DOTS   M. Boucadair, Ed.
Internet-Draft   Orange
Intended status: Standards Track   T. Reddy, Ed.
Expires: November 26, 2021   McAfee
                                                            E. Doron
                                                        Radware Ltd.
                                                             M. Chen
                                                                CMCC
                                                          J. Shallow
                                                        May 25, 2021

Distributed Denial-of-Service Open Threat Signaling (DOTS) Telemetry
draft-ietf-dots-telemetry-16

Abstract

   This document aims to enrich DOTS signal channel protocol with
   various telemetry attributes allowing optimal Distributed Denial-of-
   Service attack mitigation. It specifies the normal traffic baseline
   and attack traffic telemetry attributes a DOTS client can convey to
   its DOTS server in the mitigation request, the mitigation status
   telemetry attributes a DOTS server can communicate to a DOTS client,
   and the mitigation efficacy telemetry attributes a DOTS client can
   communicate to a DOTS server. The telemetry attributes can assist
   the mitigator to choose the DDoS mitigation techniques and perform
   optimal DDoS attack mitigation.

Copyright Notice

Table of Contents

1. Introduction

   Distributed Denial of Service (DDoS) attacks have become more
   sophisticated. IT organizations and service providers are facing
   DDoS attacks that fall into two broad categories:

   1. Network/Transport layer attacks target the victim's
       infrastructure. These attacks are not necessarily aimed at
       taking down the actual delivered services, but rather to
       eliminate various network elements (routers, switches, firewalls,
       transit links, and so on) from serving legitimate users traffic.

The main method of such attacks is to send a large volume or high
packet per second (pps) of traffic toward the victim's
infrastructure. Typically, attack volumes may vary from a few
100 Mbps to 100s of Gbps or even Tbps. Attacks are commonly
carried out leveraging botnets and attack reflectors for
amplification attacks such as NTP (Network Time Protocol), DNS
(Domain Name System), SNMP (Simple Network Management Protocol),
or SSDP (Simple Service Discovery Protocol).

2. Application layer attacks target various applications. Typical
examples include attacks against HTTP/HTTPS, DNS, SIP (Session
Initiation Protocol), or SMTP (Simple Mail Transfer Protocol).
However, all applications with their port numbers open at network
edges can be attractive attack targets.

Application layer attacks are considered more complex and hard to
categorize, therefore harder to detect and mitigate efficiently.

To compound the problem, attackers also leverage multi-vectored
attacks. These attacks are assembled from dynamic attack vectors
(Network/Application) and tactics. As such, multiple attack vectors
formed by multiple attack types and volumes are launched
simultaneously towards a victim. Multi-vector attacks are harder to
detect and defend. Multiple and simultaneous mitigation techniques
are needed to defeat such attack campaigns. It is also common for
attackers to change attack vectors right after a successful
mitigation, burdening their opponents with changing their defense
methods.

The conclusion derived from these real scenarios is that modern
attacks detection and mitigation are most certainly complicated and
highly convoluted tasks. They demand a comprehensive knowledge of
the attack attributes, the targeted normal behavior (including,
normal traffic patterns), as well as the attacker's ongoing and past
actions. Even more challenging, retrieving all the analytics needed
for detecting these attacks is not simple to obtain with the
industry's current capabilities.

The DOTS signal channel protocol [I-D.ietf-dots-rfc8782-bis] is used
to carry information about a network resource or a network (or a part
thereof) that is under a DDoS attack. Such information is sent by a
DOTS client to one or multiple DOTS servers so that appropriate
mitigation actions are undertaken on traffic deemed suspicious.
Various use cases are discussed in [I-D.ietf-dots-use-cases].

DOTS clients can be integrated within a DDoS attack detector, or
network and security elements that have been actively engaged with
ongoing attacks. The DOTS client mitigation environment determines

that it is no longer possible or practical for it to handle these
attacks. This can be due to a lack of resources or security
capabilities, as derived from the complexities and the intensity of
these attacks. In this circumstance, the DOTS client has invaluable
knowledge about the actual attacks that need to be handled by its
DOTS server(s). By enabling the DOTS client to share this
comprehensive knowledge of an ongoing attack under specific
circumstances, the DOTS server can drastically increase its ability
to accomplish successful mitigation. While the attack is being
handled by the DOTS server associated mitigation resources, the DOTS
server has the knowledge about the ongoing attack mitigation. The
DOTS server can share this information with the DOTS client so that
the client can better assess and evaluate the actual mitigation
realized.

DOTS clients can send mitigation hints derived from attack details to
DOTS servers, with the full understanding that the DOTS server may
ignore mitigation hints, as described in [RFC8612] (Gen-004).
Mitigation hints will be transmitted across the DOTS signal channel,
as the data channel may not be functional during an attack. How a
DOTS server is handling normal and attack traffic attributes, and
mitigation hints is implementation specific.

Both DOTS clients and servers can benefit this information by
presenting various information in relevant management, reporting, and
portal systems.

This document defines DOTS telemetry attributes that can be conveyed
by DOTS clients to DOTS servers, and vice versa. The DOTS telemetry
attributes are not mandatory attributes of the DOTS signal channel
protocol [I-D.ietf-dots-rfc8782-bis]. Nevertheless, when DOTS
telemetry attributes are available to a DOTS agent, and absent any
policy, it can signal the attributes in order to optimize the overall
mitigation service provisioned using DOTS.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in BCP
14 [RFC2119][RFC8174] when, and only when, they appear in all
capitals, as shown here.

The reader should be familiar with the terms defined in [RFC8612].

"DOTS Telemetry" is defined as the collection of attributes that are
used to characterize normal traffic baseline, attacks and their
mitigation measures, and any related information that may help in

   enforcing countermeasures. The DOTS Telemetry is an optional set of
   attributes that can be signaled in the DOTS signal channel protocol.

   Telemetry Setup Identifier (tsid) is an identifier that is generated
   by DOTS clients to uniquely identify DOTS telemetry setup
   configuration data.

   Telemetry Identifier (tmid) is an identifier that is generated by
   DOTS clients to uniquely identify DOTS telemetry data that is
   communicated prior or during a mitigation.

   When two telemetry requests overlap, "overlapped" lower numeric
   'tsid' (or 'tmid')" refers to the lower 'tsid' (or 'tmid') value of
   these overlapping requests.

   The meaning of the symbols in YANG tree diagrams are defined in
   [RFC8340] and [RFC8791].

3. DOTS Telemetry: Overview and Purpose

   Timely and effective signaling of up-to-date DDoS telemetry to all
   elements involved in the mitigation process is essential and improves
   the overall DDoS mitigation service effectiveness. Bi-directional
   feedback between DOTS agents is required for an increased awareness
   of each party, supporting superior and highly efficient attack
   mitigation service.

3.1. Need More Visibility

   When signaling a mitigation request, it is most certainly beneficial
   for DOTS clients to signal to DOTS servers any knowledge regarding
   ongoing attacks. This can happen in cases where DOTS clients are
   asking DOTS servers for support in defending against attacks that
   they have already detected and/or mitigated.

   If attacks are already detected and categorized within a DOTS client
   domain, the DOTS server, and its associated mitigation services, can
   proactively benefit this information and optimize the overall service
   delivery. It is important to note that DOTS client domains and DOTS
   server domains detection and mitigation approaches can be different,
   and can potentially outcome different results and attack
   classifications. The DDoS mitigation service treats the ongoing
   attack details received from DOTS clients as hints and cannot
   completely rely or trust the attack details conveyed by DOTS clients.

   A basic requirement of security operation teams is to be aware and
   get visibility into the attacks they need to handle. The DOTS server
   security operation teams benefit from the DOTS telemetry, especially

from the reports of ongoing attacks. Even if some mitigation can be
automated, operational teams can use the DOTS telemetry to be
prepared for attack mitigation and to assign the correct resources
(operation staff, networking and mitigation) for the specific
service. Similarly, security operation personnel at the DOTS client
side ask for feedback about their requests for protection.
Therefore, it is valuable for DOTS servers to share DOTS telemetry
with DOTS clients.

Mutual sharing of information is thus crucial for "closing the
mitigation loop" between DOTS clients and servers. For the server
side team, it is important to realize that the same attacks that the
DOTS server's mitigation resources are seeing are those that a DOTS
client is asking to mitigate. For the DOTS client side team, it is
important to realize that the DOTS clients receive the required
service. For example, understanding that "I asked for mitigation of
two attacks and my DOTS server detects and mitigates only one of
them". Cases of inconsistency in attack classification between DOTS
clients and servers can be highlighted, and maybe handled, using the
DOTS telemetry attributes.

In addition, management and orchestration systems, at both DOTS
client and server sides, can use DOTS telemetry as a feedback to
automate various control and management activities derived from
signaled telemetry information.

If the DOTS server's mitigation resources have the capabilities to
facilitate the DOTS telemetry, the DOTS server adapts its protection
strategy and activates the required countermeasures immediately
(automation enabled) for the sake of optimized attack mitigation
decisions and actions.

3.2. Enhanced Detection

DOTS telemetry can also be used to tune the DDoS mitigators with the
correct state of an attack. During the last few years, DDoS attack
detection technologies have evolved from threshold-based detection
(that is, cases when all or specific parts of traffic cross a
predefined threshold for a certain period of time is considered as an
attack) to an "anomaly detection" approach. For the latter, it is
required to maintain rigorous learning of "normal" behavior and where
an "anomaly" (or an attack) is identified and categorized based on
the knowledge about the normal behavior and a deviation from this
normal behavior. Machine learning approaches are used such that the
actual traffic thresholds are automatically calculated by learning
the protected entity normal traffic behavior during idle time. The
normal traffic characterization learned is referred to as the "normal

traffic baseline". An attack is detected when the victim's actual
traffic is deviating from this normal baseline.

In addition, subsequent activities toward mitigating an attack are
much more challenging. The ability to distinguish legitimate traffic
from attacker traffic on a per packet basis is complex. For example,
a packet may look "legitimate" and no attack signature can be
identified. The anomaly can be identified only after detailed
statistical analysis. DDoS attack mitigators use the normal baseline
during the mitigation of an attack to identify and categorize the
expected appearance of a specific traffic pattern. Particularly, the
mitigators use the normal baseline to recognize the "level of
normality" needs to be achieved during the various mitigation
process.

Normal baseline calculation is performed based on continuous learning
of the normal behavior of the protected entities. The minimum
learning period varies from hours to days and even weeks, depending
on the protected application behavior. The baseline cannot be
learned during active attacks because attack conditions do not
characterize the protected entities' normal behavior.

If the DOTS client has calculated the normal baseline of its
protected entities, signaling such information to the DOTS server
along with the attack traffic levels is significantly valuable. The
DOTS server benefits from this telemetry by tuning its mitigation
resources with the DOTS client's normal baseline. The DOTS server
mitigators use the baseline to familiarize themselves with the attack
victim's normal behavior and target the baseline as the level of
normality they need to achieve. Fed with this information, the
overall mitigation performances is expected to be improved in terms
of time to mitigate, accuracy, false-negative, and false-positive.

Mitigation of attacks without having certain knowledge of normal
traffic can be inaccurate at best. This is especially true for
recursive signaling (see Section 3.2.3 in [I-D.ietf-dots-use-cases]).
In addition, the highly diverse types of use cases where DOTS clients
are integrated also emphasize the need for knowledge of each DOTS
client domain behavior. Consequently, common global thresholds for
attack detection practically cannot be realized. Each DOTS client
domain can have its own levels of traffic and normal behavior.
Without facilitating normal baseline signaling, it may be very
difficult for DOTS servers in some cases to detect and mitigate the
attacks accurately:

    It is important to emphasize that it is practically impossible for
    the DOTS server's mitigators to calculate the normal baseline in

cases where they do not have any knowledge of the traffic
beforehand.

In addition, baseline learning requires a period of time that
cannot be afforded during active attack.

Of course, this information can provided using out-of-band
mechanisms or manual configuration at the risk to maintain
inaccurate information as the network evolves and "normal"
patterns change. The use of a dynamic and collaborative means
between the DOTS client and server to identify and share key
parameters for the sake of efficient DDoS protection is valuable.

3.3. Efficient Mitigation

During a high volume attack, DOTS client pipes can be totally
saturated. DOTS clients ask their DOTS servers to handle the attack
upstream so that DOTS client pipes return to a reasonable load level
(normal pattern, ideally). At this point, it is essential to ensure
that the mitigator does not overwhelm the DOTS client pipes by
sending back "clean traffic", or what it believes is "clean". This
can happen when the mitigator has not managed to detect and mitigate
all the attacks launched towards the DOTS client domain.

In this case, it can be valuable to DOTS clients to signal to DOTS
servers the "total pipe capacity", which is the level of traffic the
DOTS client domain can absorb from its upstream network. Dynamic
updates of the condition of pipes between DOTS agents while they are
under a DDoS attack is essential (e.g., where multiple DOTS clients
share the same physical connectivity pipes). It is important to note
that the term "pipe" noted here does not necessary represent physical
pipe, but rather represents the maximum level of traffic that the
DOTS client domain can receive. The DOTS server should activate
other mechanisms to ensure it does not allow the DOTS client domain's
pipes to be saturated unintentionally. The rate-limit action defined
in [RFC8783] is a reasonable candidate to achieve this objective; the
DOTS client can ask for the type(s) of traffic (such as ICMP, UDP,
TCP port number 80) it prefers to limit. The rate-limit action can
be controlled via the signal channel
[I-D.ietf-dots-signal-filter-control] even when the pipe is
overwhelmed.

4. Design Overview

4.1. Overview of Telemetry Operations

   This document specifies an extension to the DOTS signal channel
   protocol. Considerations about how to establish, maintain, and make
   use of the DOTS signal channel are specified in
   [I-D.ietf-dots-rfc8782-bis].

   Once the DOTS signal channel is established, DOTS clients that
   support the DOTS telemetry extension proceed with the telemetry setup
   configuration (e.g., measurement interval, telemetry notification
   interface, pipe capacity, normal traffic baseline) as detailed in
   Section 6. DOTS agents can then include DOTS telemetry attributes
   using the DOTS signal channel (Section 7.1). A DOTS client can use
   separate messages to share with its DOTS server(s) a set of telemetry
   data bound to an ongoing mitigation (Section 7.2). Also, a DOTS
   client that is interested to receive telemetry notifications related
   to some of its resources follows the procedure defined in
   Section 7.3. The DOTS client can then decide to send a mitigation
   request if the notified attack cannot be mitigated locally within the
   DOTS client domain.

   Aggregate DOTS telemetry data can also be included in efficacy update
   (Section 8.1) or mitigation update (Section 8.2) messages.

4.2. Generic Considerations

4.2.1. DOTS Client Identification

   Following the rules in Section 4.4.1 of [I-D.ietf-dots-rfc8782-bis],
   a unique identifier is generated by a DOTS client to prevent request
   collisions ('cuid').

   As a reminder, [I-D.ietf-dots-rfc8782-bis] forbids 'cuid' to be
   returned in a response message body.

4.2.2. DOTS Gateways

   DOTS gateways may be located between DOTS clients and servers. The
   considerations elaborated in Section 4.4.1 of
   [I-D.ietf-dots-rfc8782-bis] must be followed. In particular, 'cdid'
   attribute is used to unambiguously identify a DOTS client domain.

   As a reminder, [I-D.ietf-dots-rfc8782-bis] forbids 'cdid' (if
   present) to be returned in a response message body.

4.2.3. Empty URI Paths

Uri-Path parameters and attributes with empty values MUST NOT be
present in a request and render an entire message invalid.

4.2.4. Controlling Configuration Data

The DOTS server follows the same considerations discussed in
Section of 4.5.3 of [I-D.ietf-dots-rfc8782-bis] for managing DOTS
telemetry configuration freshness and notification.

Likewise, a DOTS client may control the selection of configuration
and non-configuration data nodes when sending a GET request by means
of the 'c' Uri-Query option and following the procedure specified in
Section of 4.4.2 of [I-D.ietf-dots-rfc8782-bis]. These
considerations are not reiterated in the following sections.

4.3. Block-wise Transfer

DOTS clients can use block wise transfer [RFC7959] with the
recommendation detailed in Section 4.4.2 of
[I-D.ietf-dots-rfc8782-bis] to control the size of a response when
the data to be returned does not fit within a single datagram.

DOTS clients can also use CoAP Block1 Option in a PUT request (see
Section 2.5 of [RFC7959]) to initiate large transfers, but these
Block1 transfers will fail if the inbound "pipe" is running full, so
consideration needs to be made to try to fit this PUT into a single
transfer, or to separate out the PUT into several discrete PUTs where
each of them fits into a single packet.

Q-Block1 and Q-Block2 Options that are similar to the CoAP Block1 and
Block2 Options, but enable robust transmissions of big blocks of data
with less packet interchanges using NON messages, are defined in
[I-D.ietf-core-new-block]. DOTS implementations can consider the use
of Q-Block1 and Q-Block2 Options.

4.4. DOTS Multi-homing Considerations

Multi-homed DOTS clients are assumed to follow the recommendations in
[I-D.ietf-dots-multihoming] to select which DOTS server to contact
and which IP prefixes to include in a telemetry message to a given
peer DOTS server. For example, if each upstream network exposes a
DOTS server and the DOTS client maintains DOTS channels with all of
them, only the information related to prefixes assigned by an
upstream network to the DOTS client domain will be signaled via the
DOTS channel established with the DOTS server of that upstream
network.

Considerations related to whether (and how) a DOTS client gleans some
telemetry information (e.g., attack details) it receives from a first
DOTS server and share it with a second DOTS server are implementation
and deployment specific.

4.5. YANG Considerations

Telemetry messages exchanged between DOTS agents are serialized using
Concise Binary Object Representation (CBOR) [RFC8949]. CBOR-encoded
payloads are used to carry signal channel specific payload messages
which convey request parameters and response information such as
errors.

This document specifies a YANG module [RFC7950] for representing DOTS
telemetry message types (Section 10.1). All parameters in the
payload of the DOTS signal channel are mapped to CBOR types as
specified in Section 11. As a reminder, Section 3 of
[I-D.ietf-dots-rfc8782-bis] defines the rules for mapping YANG-
modeled data to CBOR.

The DOTS telemetry module (Section 10.1) is not intended to be used
via NETCONF/RESTCONF for DOTS server management purposes. It serves
only to provide a data model and encoding following [RFC8791].
Server deviations are strongly discouraged as the peer DOTS agent
does not have means to retrieve the list of deviations and that
interoperability issues are likely to be encountered.

The DOTS telemetry module (Section 10.1) uses "enumerations" rather
than "identities" to define units, samples, and intervals because
otherwise the namespace identifier "ietf-dots-telemetry" must be
included when a telemetry attribute is included (e.g., in a
mitigation efficacy update). The use of "identities" is thus
suboptimal from a message compactness standpoint; one of the key
requirements for DOTS messages.

The DOTS telemetry module (Section 10.1) includes some lists for
which no key statement is included. This behavior is compliant with
[RFC8791]. The reason for not including these keys is because they
are not included in the request message body but as mandatory Uri-
Paths in requests (Sections 6 and 7). Otherwise, whenever a key
statement is used in the module, the same definition as in
Section 7.8.2 of [RFC7950] is assumed.

In order to optimize the data exchanged over the DOTS signal channel,
the document specifies a second YANG module ("ietf-dots-mapping",
Section 10.2) that augments the DOTS data channel [RFC8783]. This
augmentation can be used during idle time to share the attack mapping
details (Section 7.1.5). DOTS clients can use tools, e.g., YANG

Library [RFC8525], to retrieve the list of features and deviations
supported by the DOTS server.

4.6. A Note About Examples

Examples are provided for illustration purposes. The document does
not aim to provide a comprehensive list of message examples.

The authoritative reference for validating telemetry messages
exchanged over the DOTS signal channel are sections 6, 7, and 8
together with the mapping table established in Section 11. The
structure of telemetry message bodies is represented as a YANG data
structure (Section 10.1).

5. Telemetry Operation Paths

As discussed in Section 4.2 of [I-D.ietf-dots-rfc8782-bis], each DOTS
operation is indicated by a path suffix that indicates the intended
operation. The operation path is appended to the path prefix to form
the URI used with a CoAP request to perform the desired DOTS
operation. The following telemetry path suffixes are defined
(Table 1):

```
        +-----------------+----------------+-----------+
        | Operation       | Operation Path | Details   |
        +=================+================+===========+
        | Telemetry Setup | /tm-setup      | Section 6 |
        | Telemetry       | /tm            | Section 7 |
        +-----------------+----------------+-----------+
```

                  Table 1: DOTS Telemetry Operations

Consequently, the "ietf-dots-telemetry" YANG module defined in
Section 10.1 defines data structure to represent new DOTS message
types called 'telemetry-setup' and 'telemetry'. The tree structure
is shown in Figure 1. More details are provided in Sections 6 and 7
about the exact structure of 'telemetry-setup' and 'telemetry'
message types.

```
   structure dots-telemetry:
     +-- (telemetry-message-type)?
        +--:(telemetry-setup)
        |  ...
        |  +-- telemetry* []
        |      ...
        |     +-- (setup-type)?
        |        +--:(telemetry-config)
        |        |  ...
        |        +--:(pipe)
        |        |  ...
        |        +--:(baseline)
        |           ...
        +--:(telemetry)
           ...
```

           Figure 1: New DOTS Message Types (YANG Tree Structure)

6. DOTS Telemetry Setup Configuration

   In reference to Figure 1, a DOTS telemetry setup message MUST include
   only telemetry-related configuration parameters (Section 6.1) or
   information about DOTS client domain pipe capacity (Section 6.2) or
   telemetry traffic baseline (Section 6.3). As such, requests that
   include a mix of telemetry configuration, pipe capacity, or traffic
   baseline MUST be rejected by DOTS servers with a 4.00 (Bad Request).

   A DOTS client can reset all installed DOTS telemetry setup
   configuration data following the considerations detailed in
   Section 6.4.

   A DOTS server may detect conflicts when processing requests related
   to DOTS client domain pipe capacity or telemetry traffic baseline
   with requests from other DOTS clients of the same DOTS client domain.
   More details are included in Section 6.5.

   Telemetry setup configuration is bound to a DOTS client domain. DOTS
   servers MUST NOT expect DOTS clients to send regular requests to
   refresh the telemetry setup configuration. Any available telemetry
   setup configuration has a validity timeout of the DOTS association
   with a DOTS client domain. DOTS servers MUST NOT reset 'tsid'
   because a session failed with a DOTS client. DOTS clients update
   their telemetry setup configuration upon change of a parameter that
   may impact attack mitigation.

   DOTS telemetry setup configuration request and response messages are
   marked as Confirmable messages (Section 2.1 of [RFC7252]).

6.1. Telemetry Configuration

A DOTS client can negotiate with its server(s) a set of telemetry
configuration parameters to be used for telemetry. Such parameters
include:

o Percentile-related measurement parameters

o Measurement units

o Acceptable percentile values

o Telemetry notification interval

o Acceptable Server-originated telemetry

Section 11.3 of [RFC2330] includes more details about computing
percentiles.

6.1.1. Retrieve Current DOTS Telemetry Configuration

A GET request is used to obtain acceptable and current telemetry
configuration parameters on the DOTS server. This request may
include a 'cdid' Uri-Path when the request is relayed by a DOTS
gateway. An example of such request is depicted in Figure 2.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

     Figure 2: GET to Retrieve Current and Acceptable DOTS Telemetry
                              Configuration

Upon receipt of such request, and assuming no error is encountered by
processing the request, the DOTS server replies with a 2.05 (Content)
response that conveys the current and telemetry parameters acceptable
by the DOTS server. The tree structure of the response message body
is provided in Figure 3. Note that the response also includes any
pipe (Section 6.2) and baseline information (Section 6.3) maintained
by the DOTS server for this DOTS client.

DOTS servers that support the capability of sending telemetry
information to DOTS clients prior or during a mitigation
(Section 8.2) sets 'server-originated-telemetry' under 'max-config-
values' to 'true' ('false' is used otherwise). If 'server-
originated-telemetry' is not present in a response, this is

equivalent to receiving a request with 'server-originated-telemetry'
set to 'false'.

```
structure dots-telemetry:
  +-- (telemetry-message-type)?
     +--:(telemetry-setup)
     |  +-- (direction)?
     |  |  +--:(server-to-client-only)
     |  |     +-- max-config-values
     |  |     |  +-- measurement-interval?        interval
     |  |     |  +-- measurement-sample?          sample
     |  |     |  +-- low-percentile?              percentile
     |  |     |  +-- mid-percentile?              percentile
     |  |     |  +-- high-percentile?             percentile
     |  |     |  +-- server-originated-telemetry?  boolean
     |  |     |  +-- telemetry-notify-interval?   uint32
     |  |     +-- min-config-values
     |  |     |  +-- measurement-interval?        interval
     |  |     |  +-- measurement-sample?          sample
     |  |     |  +-- low-percentile?              percentile
     |  |     |  +-- mid-percentile?              percentile
     |  |     |  +-- high-percentile?             percentile
     |  |     |  +-- telemetry-notify-interval?   uint32
     |  |     +-- supported-unit-classes
     |  |     |  +-- unit-config* [unit]
     |  |     |     +-- unit            unit-class
     |  |     |     +-- unit-status     boolean
     |  |     +-- query-type*          query-type
     |  +-- telemetry* []
     |     +-- (direction)?
     |     |  +--:(server-to-client-only)
     |     |     +-- tsid?                   uint32
     |     +-- (setup-type)?
     |        +--:(telemetry-config)
     |        |  +-- current-config
     |        |     +-- measurement-interval?        interval
     |        |     +-- measurement-sample?          sample
     |        |     +-- low-percentile?              percentile
     |        |     +-- mid-percentile?              percentile
     |        |     +-- high-percentile?             percentile
     |        |     +-- unit-config* [unit]
     |        |     |  +-- unit            unit-class
     |        |     |  +-- unit-status     boolean
     |        |     +-- server-originated-telemetry?  boolean
     |        |     +-- telemetry-notify-interval?   uint32
     |        +--:(pipe)
     |        |  ...
     |        +--:(baseline)
```

```
            |              ...
        +--:(telemetry)
            ...
```

              Figure 3: Telemetry Configuration Tree Structure

   When both 'min-config-values' and 'max-config-values' attributes are
   present, the values carried in 'max-config-values' attributes MUST be
   greater or equal to their counterpart in 'min-config-values'
   attributes.

6.1.2. Convey DOTS Telemetry Configuration

   PUT request is used to convey the configuration parameters for the
   telemetry data (e.g., low, mid, or high percentile values). For
   example, a DOTS client may contact its DOTS server to change the
   default percentile values used as baseline for telemetry data.
   Figure 3 lists the attributes that can be set by a DOTS client in
   such PUT request. An example of a DOTS client that modifies all
   percentile reference values is shown in Figure 4.

```
   Header: PUT (Code=0.03)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "tm-setup"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "tsid=123"
   Content-Format: "application/dots+cbor"

   {
     "ietf-dots-telemetry:telemetry-setup": {
       "telemetry": [
         {
           "current-config": {
             "low-percentile": "5.00",
             "mid-percentile": "65.00",
             "high-percentile": "95.00"
           }
         }
       ]
     }
   }
```

          Figure 4: PUT to Convey the DOTS Telemetry Configuration

   'cuid' is a mandatory Uri-Path parameter for PUT requests.

   The following additional Uri-Path parameter is defined:

tsid: Telemetry Setup Identifier is an identifier for the DOTS
    telemetry setup configuration data represented as an integer.
    This identifier MUST be generated by DOTS clients. 'tsid'
    values MUST increase monotonically (when a new PUT is generated
    by a DOTS client to convey new configuration parameters for the
    telemetry).

    The procedure specified in Section 4.4.1 of
    [I-D.ietf-dots-rfc8782-bis] MUST be followed for 'tsid'
    rollover.

    This is a mandatory attribute. 'tsid' MUST follow 'cuid'.

'cuid' and 'tsid' MUST NOT appear in the PUT request message body.

At least one configurable attribute MUST be present in the PUT
request.

The PUT request with a higher numeric 'tsid' value overrides the DOTS
telemetry configuration data installed by a PUT request with a lower
numeric 'tsid' value. To avoid maintaining a long list of 'tsid'
requests for requests carrying telemetry configuration data from a
DOTS client, the lower numeric 'tsid' MUST be automatically deleted
and no longer be available at the DOTS server.

The DOTS server indicates the result of processing the PUT request
using the following Response Codes:

o If the request is missing a mandatory attribute, does not include
   'cuid' or 'tsid' Uri-Path parameters, or contains one or more
   invalid or unknown parameters, 4.00 (Bad Request) MUST be returned
   in the response.

o If the DOTS server does not find the 'tsid' parameter value
   conveyed in the PUT request in its configuration data and if the
   DOTS server has accepted the configuration parameters, then a 2.01
   (Created) Response Code MUST be returned in the response.

o If the DOTS server finds the 'tsid' parameter value conveyed in
   the PUT request in its configuration data and if the DOTS server
   has accepted the updated configuration parameters, 2.04 (Changed)
   MUST be returned in the response.

o If any of the enclosed configurable attribute values are not
   acceptable to the DOTS server (Section 6.1.1), 4.22 (Unprocessable
   Entity) MUST be returned in the response.

   The DOTS client may retry and send the PUT request with updated
   attribute values acceptable to the DOTS server.

By default, low percentile (10th percentile), mid percentile (50th
percentile), high percentile (90th percentile), and peak (100th
percentile) values are used to represent telemetry data.
Nevertheless, a DOTS client can disable some percentile types (low,
mid, high). In particular, setting 'low-percentile' to '0.00'
indicates that the DOTS client is not interested in receiving low-
percentiles. Likewise, setting 'mid-percentile' (or 'high-
percentile') to the same value as 'low-percentile' (or 'mid-
percentile') indicates that the DOTS client is not interested in
receiving mid-percentiles (or high-percentiles). For example, a DOTS
client can send the request depicted in Figure 5 to inform the server
that it is interested in receiving only high-percentiles. This
assumes that the client will only use that percentile type when
sharing telemetry data with the server.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=569"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "current-config": {
          "low-percentile": "0.00",
          "mid-percentile": "0.00",
          "high-percentile": "95.00"
        }
      }
    ]
  }
}
```

              Figure 5: PUT to Disable Low- and Mid-Percentiles

DOTS clients can also configure the unit class(es) to be used for
traffic-related telemetry data among the following supported unit
classes: packets per second, bits per second, and bytes per second.

DOTS clients that are interested to receive pre or ongoing mitigation
telemetry (pre-or-ongoing-mitigation) information from a DOTS server

(Section 8.2) MUST set 'server-originated-telemetry' to 'true'. If
'server-originated-telemetry' is not present in a PUT request, this
is equivalent to receiving a request with 'server-originated-
telemetry' set to 'false'. An example of a request to enable pre-or-
ongoing-mitigation telemetry from DOTS servers is shown in Figure 6.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=569"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "current-config": {
          "server-originated-telemetry": true
        }
      }
    ]
  }
}
```

Figure 6: PUT to Enable Pre-or-ongoing-mitigation Telemetry from the
                             DOTS server

6.1.3. Retrieve Installed DOTS Telemetry Configuration

A DOTS client may issue a GET message with 'tsid' Uri-Path parameter
to retrieve the current DOTS telemetry configuration. An example of
such request is depicted in Figure 7.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=123"
```

Figure 7: GET to Retrieve Current DOTS Telemetry Configuration

If the DOTS server does not find the 'tsid' Uri-Path value conveyed
in the GET request in its configuration data for the requesting DOTS
client, it MUST respond with a 4.04 (Not Found) error Response Code.

6.1.4. Delete DOTS Telemetry Configuration

A DELETE request is used to delete the installed DOTS telemetry
configuration data (Figure 8). 'cuid' and 'tsid' are mandatory Uri-
Path parameters for such DELETE requests.

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=123"
```

Figure 8: Delete Telemetry Configuration

The DOTS server resets the DOTS telemetry configuration back to the
default values and acknowledges a DOTS client's request to remove the
DOTS telemetry configuration using 2.02 (Deleted) Response Code. A
2.02 (Deleted) Response Code is returned even if the 'tsid' parameter
value conveyed in the DELETE request does not exist in its
configuration data before the request.

Section 6.4 discusses the procedure to reset all DOTS telemetry setup
configuration.

6.2. Total Pipe Capacity

A DOTS client can communicate to the DOTS server(s) its DOTS client
domain pipe information. The tree structure of the pipe information
is shown in Figure 9.

```
    structure dots-telemetry:
      +-- (telemetry-message-type)?
         +--:(telemetry-setup)
         |  ...
         |  +-- telemetry* []
         |     +-- (direction)?
         |     |  +--:(server-to-client-only)
         |     |     +-- tsid?                 uint32
         |     +-- (setup-type)?
         |        +--:(telemetry-config)
         |        |  ...
         |        +--:(pipe)
         |        |  +-- total-pipe-capacity* [link-id unit]
         |        |     +-- link-id     nt:link-id
         |        |     +-- capacity    uint64
         |        |     +-- unit        unit
         |        +--:(baseline)
         |           ...
         +--:(telemetry)
            ...
```

                     Figure 9: Pipe Tree Structure

   A DOTS client domain pipe is defined as a list of limits of
   (incoming) traffic volume ('total-pipe-capacity') that can be
   forwarded over ingress interconnection links of a DOTS client domain.
   Each of these links is identified with a 'link-id' [RFC8345].

   The unit used by a DOTS client when conveying pipe information is
   captured in 'unit' attribute. The DOTS client MUST auto-scale so
   that the appropriate unit is used.

6.2.1. Convey DOTS Client Domain Pipe Capacity

   Similar considerations to those specified in Section 6.1.2 are
   followed with one exception:

      The relative order of two PUT requests carrying DOTS client domain
      pipe attributes from a DOTS client is determined by comparing
      their respective 'tsid' values. If such two requests have
      overlapping 'link-id' and 'unit', the PUT request with higher
      numeric 'tsid' value will override the request with a lower
      numeric 'tsid' value. The overlapped lower numeric 'tsid' MUST be
      automatically deleted and no longer be available.

   DOTS clients SHOULD minimize the number of active 'tsids' used for
   pipe information. In order to avoid maintaining a long list of
   'tsids' for pipe information, it is RECOMMENDED that DOTS clients

   include in any request to update information related to a given link
   the information of other links (already communicated using a lower
   'tsid' value). Doing so, this update request will override these
   existing requests and hence optimize the number of 'tsid' request per
   DOTS client.

   o Note: This assumes that all link information can fit in one single
      message.

   For example, a DOTS client managing a single homed domain (Figure 10)
   can send a PUT request (shown in Figure 11) to communicate the
   capacity of "link1" used to connect to its ISP.

```
                     ,--,--,--.              ,--,--,--.
                  ,-'          `-.        ,-'          `-.
                 (  DOTS Client   )=====(      ISP#A      )
                  `-.  Domain  ,-' link1 `-.          ,-'
                    `--'--'--'              `--'--'--'
```

                 Figure 10: Single Homed DOTS Client Domain

   Header: PUT (Code=0.03)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "tm-setup"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "tsid=457"
   Content-Format: "application/dots+cbor"

   {
     "ietf-dots-telemetry:telemetry-setup": {
       "telemetry": [
         {
           "total-pipe-capacity": [
             {
               "link-id": "link1",
               "capacity": "500",
               "unit": "megabit-ps"
             }
           ]
         }
       ]
     }
   }

        Figure 11: Example of a PUT Request to Convey Pipe Information
                              (Single Homed)

DOTS clients may be instructed to signal a link aggregate instead of
individual links. For example, a DOTS client that manages a DOTS
client domain having two interconnection links with an upstream ISP
(Figure 12) can send a PUT request (shown in Figure 13) to
communicate the aggregate link capacity with its ISP. Signalling
individual or aggregate link capacity is deployment specific.

```
                 ,--,--,--.               ,--,--,--.
               ,-'           `-.===== ,-'           `-.
              (   DOTS Client   )    (      ISP#C      )
               `-.  Domain   ,-'====== `-.         ,-'
                 `--'--'--'               `--'--'--'
```
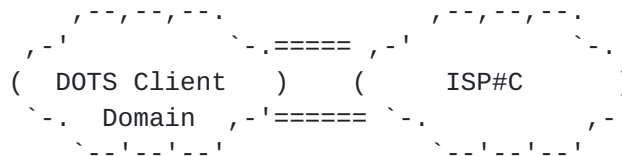
        Figure 12: DOTS Client Domain with Two Interconnection Links

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=896"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "total-pipe-capacity": [
          {
            "link-id": "aggregate",
            "capacity": "700",
            "unit": "megabit-ps"
          }
        ]
      }
    ]
  }
}
```

      Figure 13: Example of a PUT Request to Convey Pipe Information
                          (Aggregated Link)

Now consider that the DOTS client domain was upgraded to connect to
an additional ISP (e.g., ISP#B of Figure 14), the DOTS client can
inform a third-party DOTS server (that is, not hosted with ISP#A and
ISP#B domains) about this update by sending the PUT request depicted
in Figure 15. This request also includes information related to
"link1" even if that link is not upgraded. Upon receipt of this

request, the DOTS server removes the request with 'tsid=457' and
updates its configuration base to maintain two links (link#1 and
link#2).

```
                    ,--,--,--.
                  ,-'          `-.
                 (       ISP#B      )
                  `-.            ,-'
                    `--'--'--'
                        ||
                        || link2
          ,--,--,--.              ,--,--,--.
        ,-'          `-.        ,-'          `-.
       (   DOTS Client   )=====(      ISP#A       )
        `-.  Domain   ,-' link1 `-.            ,-'
          `--'--'--'              `--'--'--'
```
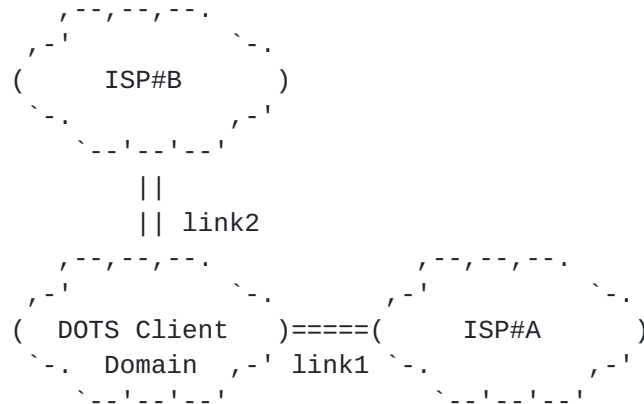
                Figure 14: Multi-Homed DOTS Client Domain

```
   Header: PUT (Code=0.03)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "tm-setup"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "tsid=458"
   Content-Format: "application/dots+cbor"

   {
     "ietf-dots-telemetry:telemetry-setup": {
       "telemetry": [
         {
           "total-pipe-capacity": [
             {
               "link-id": "link1",
               "capacity": "500",
               "unit": "megabit-ps"
             },
             {
               "link-id": "link2",
               "capacity": "500",
               "unit": "megabit-ps"
             }
           ]
         }
       ]
     }
   }
```

           Figure 15: Example of a PUT Request to Convey Pipe Information
                                (Multi-Homed)

   A DOTS client can delete a link by sending a PUT request with the
   'capacity' attribute set to "0" if other links are still active for
   the same DOTS client domain (see Section 6.2.3 for other delete
   cases). For example, if a DOTS client domain re-homes (that is, it
   changes its ISP), the DOTS client can inform its DOTS server about
   this update (e.g., from the network configuration in Figure 10 to the
   one shown in Figure 16) by sending the PUT request depicted in
   Figure 17. Upon receipt of this request, and assuming no error is
   encountered when processing the request, the DOTS server removes
   "link1" from its configuration bases for this DOTS client domain.
   Note that if the DOTS server receives a PUT request with a 'capacity'
   attribute set to "0" for all included links, it MUST reject the
   request with a 4.00 (Bad Request).

```
                      ,--,--,--.
                   ,-'          `-.
                  (      ISP#B      )
                   `-.          ,-'
                     `--'--'--'
                        ||
                        || link2
                   ,--,--,--.
                  ,-'          `-.
                 (   DOTS Client   )
                  `-.  Domain  ,-'
                    `--'--'--'
```
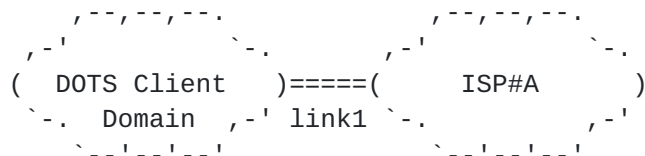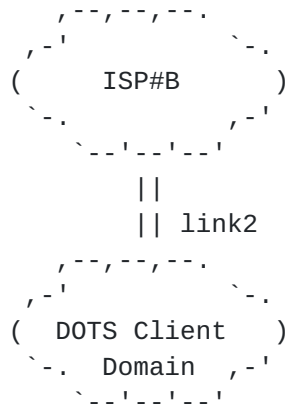
                 Figure 16: Multi-Homed DOTS Client Domain

```
    Header: PUT (Code=0.03)
    Uri-Path: ".well-known"
    Uri-Path: "dots"
    Uri-Path: "tm-setup"
    Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
    Uri-Path: "tsid=459"
    Content-Format: "application/dots+cbor"

    {
      "ietf-dots-telemetry:telemetry-setup": {
        "telemetry": [
          {
            "total-pipe-capacity": [
              {
                "link-id": "link1",
                "capacity": "0",
                "unit": "megabit-ps"
              },
              {
                "link-id": "link2",
                "capacity": "500",
                "unit": "megabit-ps"
              }
            ]
          }
        ]
      }
    }
```

        Figure 17: Example of a PUT Request to Convey Pipe Information
                              (Multi-Homed)

6.2.2. Retrieve Installed DOTS Client Domain Pipe Capacity

A GET request with 'tsid' Uri-Path parameter is used to retrieve a specific installed DOTS client domain pipe related information. The same procedure as defined in Section 6.1.3 is followed.

To retrieve all pipe information bound to a DOTS client, the DOTS client proceeds as specified in Section 6.1.1.

6.2.3. Delete Installed DOTS Client Domain Pipe Capacity

A DELETE request is used to delete the installed DOTS client domain pipe related information. The same procedure as defined in Section 6.1.4 is followed.

6.3. Telemetry Baseline

A DOTS client can communicate to its DOTS server(s) its normal traffic baseline and connections capacity:

Total traffic normal baseline: The percentile values representing the total traffic normal baseline. It can be represented for a target using 'total-traffic-normal'.

   The traffic normal per protocol ('total-traffic-normal-per-protocol') baseline is represented for a target and is transport-protocol specific.

   The traffic normal per port number ('total-traffic-normal-per-port') baseline is represented for each port number bound to a target.

   If the DOTS client negotiated percentile values and units (Section 6.1), these negotiated parameters will be used instead of the default ones. For each used unit class, the DOTS client MUST auto-scale so that the appropriate unit is used.

Total connections capacity: If the target is subjected to resource consuming DDoS attacks, the following optional attributes for the target per transport protocol are useful to detect resource consuming DDoS attacks:

   * The maximum number of simultaneous connections that are allowed to the target.

   * The maximum number of simultaneous connections that are allowed to the target per client.

   * The maximum number of simultaneous embryonic connections that
      are allowed to the target. The term "embryonic connection"
      refers to a connection whose connection handshake is not
      finished. Embryonic connection is only possible in connection-
      oriented transport protocols like TCP or SCTP.

   * The maximum number of simultaneous embryonic connections that
      are allowed to the target per client.

   * The maximum number of connections allowed per second to the
      target.

   * The maximum number of connections allowed per second to the
      target per client.

   * The maximum number of requests allowed per second to the
      target.

   * The maximum number of requests allowed per second to the target
      per client.

   * The maximum number of partial requests allowed per second to
      the target. Attacks relying upon partial requests create a
      connection with a target but do not send a complete request
      (e.g., HTTP request).

   * The maximum number of partial requests allowed per second to
      the target per client.

   The aggregate per transport protocol is captured in 'total-
   connection-capacity', while port specific capabilities are
   represented using 'total-connection-capacity-per-port'.

The tree structure of the normal traffic baseline is shown in
Figure 18.

```
 structure dots-telemetry:
   +-- (telemetry-message-type)?
     +--:(telemetry-setup)
     |  ...
     |  +-- telemetry* []
     |     +-- (direction)?
     |     |  +--:(server-to-client-only)
     |     |     +-- tsid?                    uint32
     |     +-- (setup-type)?
     |        +--:(telemetry-config)
     |        |  ...
     |        +--:(pipe)
```

```
|             |  ...
|          +--:(baseline)
|             +-- baseline* [id]
|                +-- id
|                |      uint32
|                +-- target-prefix*
|                |      inet:ip-prefix
|                +-- target-port-range* [lower-port]
|                |  +-- lower-port    inet:port-number
|                |  +-- upper-port?   inet:port-number
|                +-- target-protocol*                    uint8
|                +-- target-fqdn*
|                |      inet:domain-name
|                +-- target-uri*
|                |      inet:uri
|                +-- alias-name*
|                |      string
|                +-- total-traffic-normal* [unit]
|                |  +-- unit                unit
|                |  +-- low-percentile-g?   yang:gauge64
|                |  +-- mid-percentile-g?   yang:gauge64
|                |  +-- high-percentile-g?  yang:gauge64
|                |  +-- peak-g?             yang:gauge64
|                +-- total-traffic-normal-per-protocol*
|                |      [unit protocol]
|                |  +-- protocol           uint8
|                |  +-- unit               unit
|                |  +-- low-percentile-g?   yang:gauge64
|                |  +-- mid-percentile-g?   yang:gauge64
|                |  +-- high-percentile-g?  yang:gauge64
|                |  +-- peak-g?             yang:gauge64
|                +-- total-traffic-normal-per-port* [unit port]
|                |  +-- port               inet:port-number
|                |  +-- unit               unit
|                |  +-- low-percentile-g?   yang:gauge64
|                |  +-- mid-percentile-g?   yang:gauge64
|                |  +-- high-percentile-g?  yang:gauge64
|                |  +-- peak-g?             yang:gauge64
|                +-- total-connection-capacity* [protocol]
|                |  +-- protocol               uint8
|                |  +-- connection?             uint64
|                |  +-- connection-client?      uint64
|                |  +-- embryonic?              uint64
|                |  +-- embryonic-client?       uint64
|                |  +-- connection-ps?          uint64
|                |  +-- connection-client-ps?   uint64
|                |  +-- request-ps?             uint64
|                |  +-- request-client-ps?      uint64
```

```
        |                  |  +-- partial-request-ps?         uint64
        |                  |  +-- partial-request-client-ps?   uint64
        |                  +-- total-connection-capacity-per-port*
        |                          [protocol port]
        |                     +-- port
        |                     |      inet:port-number
        |                     +-- protocol                     uint8
        |                     +-- connection?                  uint64
        |                     +-- connection-client?           uint64
        |                     +-- embryonic?                   uint64
        |                     +-- embryonic-client?            uint64
        |                     +-- connection-ps?               uint64
        |                     +-- connection-client-ps?        uint64
        |                     +-- request-ps?                  uint64
        |                     +-- request-client-ps?           uint64
        |                     +-- partial-request-ps?          uint64
        |                     +-- partial-request-client-ps?   uint64
        +--:(telemetry)
           ...
```

                 Figure 18: Telemetry Baseline Tree Structure

6.3.1. Convey DOTS Client Domain Baseline Information

   Similar considerations to those specified in Section 6.1.2 are
   followed with one exception:

      The relative order of two PUT requests carrying DOTS client domain
      baseline attributes from a DOTS client is determined by comparing
      their respective 'tsid' values. If such two requests have
      overlapping targets, the PUT request with higher numeric 'tsid'
      value will override the request with a lower numeric 'tsid' value.
      The overlapped lower numeric 'tsid' MUST be automatically deleted
      and no longer be available.

   Two PUT requests from a DOTS client have overlapping targets if there
   is a common IP address, IP prefix, FQDN, URI, or alias-name. Also,
   two PUT requests from a DOTS client have overlapping targets if the
   addresses associated with the FQDN, URI, or alias are overlapping
   with each other or with 'target-prefix'.

   DOTS clients SHOULD minimize the number of active 'tsids' used for
   baseline information. In order to avoid maintaining a long list of
   'tsids' for baseline information, it is RECOMMENDED that DOTS clients
   include in a request to update information related to a given target,
   the information of other targets (already communicated using a lower
   'tsid' value) (assuming this fits within one single datagram). This

   update request will override these existing requests and hence
   optimize the number of 'tsid' request per DOTS client.

   If no target attribute is included in the request, this is an
   indication that the baseline information applies for the DOTS client
   domain as a whole.

   An example of a PUT request to convey the baseline information is
   shown in Figure 19.

```
   Header: PUT (Code=0.03)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "tm-setup"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "tsid=126"
   Content-Format: "application/dots+cbor"

   {
     "ietf-dots-telemetry:telemetry-setup": {
       "telemetry": [
         {
           "baseline": [
             {
               "id": 1,
               "target-prefix": [
                 "2001:db8:6401::1/128",
                 "2001:db8:6401::2/128"
               ],
               "total-traffic-normal": [
                 {
                   "unit": "megabit-ps",
                   "peak-g": "60"
                 }
               ]
             }
           ]
         }
       ]
     }
   }
```

            Figure 19: PUT to Convey the DOTS Traffic Baseline

   The DOTS client may share protocol specific baseline information
   (e.g., TCP and UDP) as shown in Figure 19.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tsid=128"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry-setup": {
    "telemetry": [
      {
        "baseline": [
          {
            "id": 1,
            "target-prefix": [
              "2001:db8:6401::1/128",
              "2001:db8:6401::2/128"
            ],
            "total-traffic-normal-per-protocol": [
              {
                "unit": "megabit-ps",
                "protocol": 6,
                "peak-g": "50"
              },
              {
                "unit": "megabit-ps",
                "protocol": 17,
                "peak-g": "10"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

         Figure 20: PUT to Convey the DOTS Traffic Baseline (2)

   The normal traffic baseline information should be updated to reflect
   legitimate overloads (e.g., flash crowds) to prevent unnecessary
   mitigation.

6.3.2. Retrieve Installed Normal Traffic Baseline

A GET request with 'tsid' Uri-Path parameter is used to retrieve a
specific installed DOTS client domain baseline traffic information.
The same procedure as defined in Section 6.1.3 is followed.

To retrieve all baseline information bound to a DOTS client, the DOTS
client proceeds as specified in Section 6.1.1.

6.3.3. Delete Installed Normal Traffic Baseline

A DELETE request is used to delete the installed DOTS client domain
normal traffic baseline. The same procedure as defined in
Section 6.1.4 is followed.

6.4. Reset Installed Telemetry Setup

Upon bootstrapping (or reboot or any other event that may alter the
DOTS client setup), a DOTS client MAY send a DELETE request to set
the telemetry parameters to default values. Such a request does not
include any 'tsid'. An example of such request is depicted in
Figure 21.

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm-setup"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

Figure 21: Delete Telemetry Configuration

6.5. Conflict with Other DOTS Clients of the Same Domain

A DOTS server may detect conflicts between requests to convey pipe
and baseline information received from DOTS clients of the same DOTS
client domain. 'conflict-information' is used to report the conflict
to the DOTS client following similar conflict handling discussed in
Section 4.4.1 of [I-D.ietf-dots-rfc8782-bis]. The conflict cause can
be set to one of these values:

   1: Overlapping targets (Section 4.4.1 of
   [I-D.ietf-dots-rfc8782-bis]).

   TBA: Overlapping pipe scope (see Section 12).

7. **DOTS Pre-or-Ongoing Mitigation Telemetry**

   There are two broad types of DDoS attacks, one is bandwidth consuming
   attack, the other is target resource consuming attack. This section
   outlines the set of DOTS telemetry attributes (Section 7.1) that
   covers both the types of attacks. The objective of these attributes
   is to allow for the complete knowledge of attacks and the various
   particulars that can best characterize attacks.

   The "ietf-dots-telemetry" YANG module (Section 10.1) defines the data
   structure of a new message type called 'telemetry'. The tree
   structure of the 'telemetry' message type is shown in Figure 24.

   The pre-or-ongoing-mitigation telemetry attributes are indicated by
   the path suffix '/tm'. The '/tm' is appended to the path prefix to
   form the URI used with a CoAP request to signal the DOTS telemetry.
   Pre-or-ongoing-mitigation telemetry attributes specified in
   Section 7.1 can be signaled between DOTS agents.

   Pre-or-ongoing-mitigation telemetry attributes may be sent by a DOTS
   client or a DOTS server.

   DOTS agents SHOULD bind pre-or-ongoing-mitigation telemetry data with
   mitigation requests relying upon the target attribute. In
   particular, a telemetry PUT request sent after a mitigation request
   may include a reference to that mitigation request ('mid-list') as
   shown in Figure 22. An example illustrating requests correlation by
   means of 'target-prefix' is shown in Figure 23.

   When generating telemetry data to send to a peer, the DOTS agent MUST
   auto-scale so that appropriate unit(s) are used.

```
   +-----------+                                     +-----------+
   |DOTS client|                                     |DOTS server|
   +-----------+                                     +-----------+
        |                                                 |
        |=========Mitigation Request (mid)===============>|
        |                                                 |
        |=============== Telemetry (mid-list{mid})=======>|
        |                                                 |
```
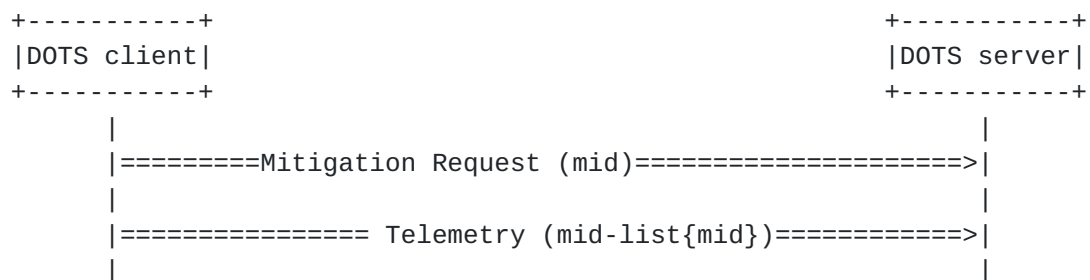
            Figure 22: Example of Request Correlation using 'mid'

```
   +-----------+                              +-----------+
   |DOTS client|                              |DOTS server|
   +-----------+                              +-----------+
        |                                           |
        |<=============== Telemetry (target-prefix)=============|
        |                                           |
        |=========Mitigation Request (target-prefix)==========>|
        |                                           |
```
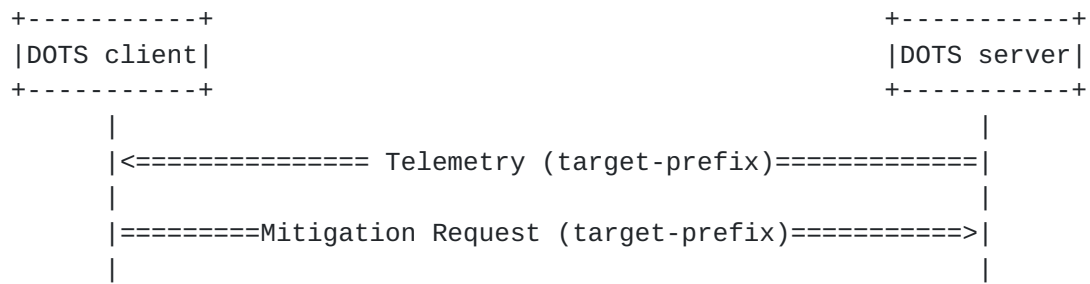
            Figure 23: Example of Request Correlation using Target Prefix

   DOTS agents MUST NOT send pre-or-ongoing-mitigation telemetry
   notifications to the same peer more frequently than once every
   'telemetry-notify-interval' (Section 6.1). If a telemetry
   notification is sent using a block-like transfer mechanism (e.g.,
   [I-D.ietf-core-new-block]), this rate limit policy MUST NOT consider
   these individual blocks as separate notifications, but as a single
   notification.

   DOTS pre-or-ongoing-mitigation telemetry request and response
   messages MUST be marked as Non-Confirmable messages (Section 2.1 of
   [RFC7252]).

```
    structure dots-telemetry:
      +-- (telemetry-message-type)?
         +--:(telemetry-setup)
         |   ...
         |   +-- telemetry* []
         |      +-- (direction)?
         |      |   +--:(server-to-client-only)
         |      |      +-- tsid?                    uint32
         |      +-- (setup-type)?
         |         +--:(telemetry-config)
         |         |   ...
         |         +--:(pipe)
         |         |   ...
         |         +--:(baseline)
         |             ...
         +--:(telemetry)
            +-- pre-or-ongoing-mitigation* []
               +-- (direction)?
               |   +--:(server-to-client-only)
               |      +-- tmid?                    uint32
               +-- target
               |   ...
               +-- total-traffic* [unit]
               |   ...
               +-- total-traffic-protocol* [unit protocol]
               |   ...
               +-- total-traffic-port* [unit port]
               |   ...
               +-- total-attack-traffic* [unit]
               |   ...
               +-- total-attack-traffic-protocol* [unit protocol]
               |   ...
               +-- total-attack-traffic-port* [unit port]
               |   ...
               +-- total-attack-connection
               |   ...
               +-- total-attack-connection-port
               |   ...
               +-- attack-detail* [vendor-id attack-id]
                   ...
```

                Figure 24: Telemetry Message Type Tree Structure

7.1. Pre-or-Ongoing-Mitigation DOTS Telemetry Attributes

   The description and motivation behind each attribute are presented in
   Section 3. DOTS telemetry attributes are optionally signaled and

therefore MUST NOT be treated as mandatory fields in the DOTS signal
channel protocol.

7.1.1. Target

A target resource (Figure 25) is identified using the attributes
'target-prefix', 'target-port-range', 'target-protocol', 'target-
fqdn', 'target-uri', 'alias-name', or a pointer to a mitigation
request ('mid-list').

```
      +--:(telemetry)
         +-- pre-or-ongoing-mitigation* []
            +-- (direction)?
            |   +--:(server-to-client-only)
            |      +-- tmid?                       uint32
            +-- target
            |   +-- target-prefix*        inet:ip-prefix
            |   +-- target-port-range* [lower-port]
            |   |   +-- lower-port     inet:port-number
            |   |   +-- upper-port?   inet:port-number
            |   +-- target-protocol*     uint8
            |   +-- target-fqdn*         inet:domain-name
            |   +-- target-uri*          inet:uri
            |   +-- alias-name*          string
            |   +-- mid-list*            uint32
            +-- total-traffic* [unit]
            |   ...
            +-- total-traffic-protocol* [unit protocol]
            |   ...
            +-- total-traffic-port* [unit port]
            |   ...
            +-- total-attack-traffic* [unit]
            |   ...
            +-- total-attack-traffic-protocol* [unit protocol]
            |   ...
            +-- total-attack-traffic-port* [unit port]
            |   ...
            +-- total-attack-connection
            |   ...
            +-- total-attack-connection-port
            |   ...
            +-- attack-detail* [vendor-id attack-id]
                ...
```

Figure 25: Target Tree Structure

   At least one of the attributes 'target-prefix', 'target-fqdn',
   'target-uri', 'alias-name', or 'mid-list' MUST be present in the
   target definition.

   If the target is subjected to bandwidth consuming attack, the
   attributes representing the percentile values of the 'attack-id'
   attack traffic are included.

   If the target is subjected to resource consuming DDoS attacks, the
   same attributes defined for Section 7.1.4 are applicable for
   representing the attack.

   This is an optional attribute.

7.1.2. Total Traffic

   The 'total-traffic' attribute (Figure 26) conveys the percentile
   values (including peak and current observed values) of total traffic
   observed during a DDoS attack. More granular total traffic can be
   conveyed in 'total-traffic-protocol' and 'total-traffic-port'.

   The 'total-traffic-protocol' represents the total traffic for a
   target and is transport-protocol specific.

   The 'total-traffic-port' represents the total traffic for a target
   per port number.

```
       +--:(telemetry)
          +-- pre-or-ongoing-mitigation* []
             +-- (direction)?
             |   +--:(server-to-client-only)
             |      +-- tmid?                      uint32
             +-- target
             |  ...
             +-- total-traffic* [unit]
             |  +-- unit                unit
             |  +-- low-percentile-g?     yang:gauge64
             |  +-- mid-percentile-g?     yang:gauge64
             |  +-- high-percentile-g?    yang:gauge64
             |  +-- peak-g?               yang:gauge64
             |  +-- current-g?            yang:gauge64
             +-- total-traffic-protocol* [unit protocol]
             |  +-- protocol             uint8
             |  +-- unit                 unit
             |  +-- low-percentile-g?     yang:gauge64
             |  +-- mid-percentile-g?     yang:gauge64
             |  +-- high-percentile-g?    yang:gauge64
             |  +-- peak-g?               yang:gauge64
             |  +-- current-g?            yang:gauge64
             +-- total-traffic-port* [unit port]
             |  +-- port                 inet:port-number
             |  +-- unit                 unit
             |  +-- low-percentile-g?     yang:gauge64
             |  +-- mid-percentile-g?     yang:gauge64
             |  +-- high-percentile-g?    yang:gauge64
             |  +-- peak-g?               yang:gauge64
             |  +-- current-g?            yang:gauge64
             +-- total-attack-traffic* [unit]
             |  ...
             +-- total-attack-traffic-protocol* [unit protocol]
             |  ...
             +-- total-attack-traffic-port* [unit port]
             |  ...
             +-- total-attack-connection
             |  ...
             +-- total-attack-connection-port
             |  ...
             +-- attack-detail* [vendor-id attack-id]
                ...
```

                 Figure 26: Total Traffic Tree Structure

7.1.3. Total Attack Traffic

   The 'total-attack-traffic' attribute (Figure 27) conveys the total
   attack traffic identified by the DOTS client domain's DDoS Mitigation
   System (or DDoS Detector). More granular total traffic can be
   conveyed in 'total-attack-traffic-protocol' and 'total-attack-
   traffic-port'.

   The 'total-attack-traffic-protocol' represents the total attack
   traffic for a target and is transport-protocol specific.

   The 'total-attack-traffic-port' represents the total attack traffic
   for a target per port number.

```
          +--:(telemetry)
             +-- pre-or-ongoing-mitigation* []
                +-- (direction)?
                |   +--:(server-to-client-only)
                |      +-- tmid?                      uint32
                +-- target
                |  ...
                +-- total-traffic* [unit]
                |  ...
                +-- total-traffic-protocol* [unit protocol]
                |  ...
                +-- total-traffic-port* [unit port]
                |  ...
                +-- total-attack-traffic* [unit]
                |  +-- protocol?          uint8
                |  +-- unit               unit
                |  +-- low-percentile-g?    yang:gauge64
                |  +-- mid-percentile-g?    yang:gauge64
                |  +-- high-percentile-g?   yang:gauge64
                |  +-- peak-g?              yang:gauge64
                |  +-- current-g?           yang:gauge64
                +-- total-attack-traffic-protocol* [unit protocol]
                |  +-- protocol           uint8
                |  +-- unit               unit
                |  +-- low-percentile-g?    yang:gauge64
                |  +-- mid-percentile-g?    yang:gauge64
                |  +-- high-percentile-g?   yang:gauge64
                |  +-- peak-g?              yang:gauge64
                |  +-- current-g?           yang:gauge64
                +-- total-attack-traffic-port* [unit port]
                |  +-- port               inet:port-number
                |  +-- unit               unit
                |  +-- low-percentile-g?    yang:gauge64
                |  +-- mid-percentile-g?    yang:gauge64
                |  +-- high-percentile-g?   yang:gauge64
                |  +-- peak-g?              yang:gauge64
                |  +-- current-g?           yang:gauge64
                +-- total-attack-connection
                |  ...
                +-- total-attack-connection-port
                |  ...
                +-- attack-detail* [vendor-id attack-id]
                   ...
```

                Figure 27: Total Attack Traffic Tree Structure

7.1.4. Total Attack Connections

   If the target is subjected to resource consuming DDoS attack, the
   'total-attack-connection' attribute is used to convey the percentile
   values (including peak and current observed values) of total attack
   connections. The following optional subattributes for the target per
   transport protocol are included to represent the attack
   characteristics:

   o The number of simultaneous attack connections to the target.
   o The number of simultaneous embryonic connections to the target.
   o The number of attack connections per second to the target.
   o The number of attack requests to the target.

   The total attack connections per port number is represented using
   'total-attack-connection-port' attribute.

```
        +--:(telemetry)
           +-- pre-or-ongoing-mitigation* []
              +-- (direction)?
              |  +--:(server-to-client-only)
              |     +-- tmid?                        uint32
              +-- target
              |  ...
              +-- total-traffic* [unit]
              |  ...
              +-- total-traffic-protocol* [unit protocol]
              |  ...
              +-- total-traffic-port* [unit port]
              |  ...
              +-- total-attack-traffic* [unit]
              |  ...
              +-- total-attack-traffic-protocol* [unit protocol]
              |  ...
              +-- total-attack-traffic-port* [unit port]
              |  ...
              +-- total-attack-connection
              |  +-- low-percentile-l* [protocol]
              |  |  +-- protocol             uint8
              |  |  +-- connection?          yang:gauge64
              |  |  +-- embryonic?           yang:gauge64
              |  |  +-- connection-ps?       yang:gauge64
              |  |  +-- request-ps?          yang:gauge64
              |  |  +-- partial-request-ps?  yang:gauge64
              |  +-- mid-percentile-l* [protocol]
              |  |  +-- protocol             uint8
              |  |  +-- connection?          yang:gauge64
              |  |  +-- embryonic?           yang:gauge64
```

```
|  |  +-- connection-ps?       yang:gauge64
|  |  +-- request-ps?          yang:gauge64
|  |  +-- partial-request-ps?  yang:gauge64
|  +-- high-percentile-l* [protocol]
|  |  +-- protocol             uint8
|  |  +-- connection?          yang:gauge64
|  |  +-- embryonic?           yang:gauge64
|  |  +-- connection-ps?       yang:gauge64
|  |  +-- request-ps?          yang:gauge64
|  |  +-- partial-request-ps?  yang:gauge64
|  +-- peak-l* [protocol]
|  |  +-- protocol             uint8
|  |  +-- connection?          yang:gauge64
|  |  +-- embryonic?           yang:gauge64
|  |  +-- connection-ps?       yang:gauge64
|  |  +-- request-ps?          yang:gauge64
|  |  +-- partial-request-ps?  yang:gauge64
|  +-- current-l* [protocol]
|     +-- protocol             uint8
|     +-- connection?          yang:gauge64
|     +-- embryonic?           yang:gauge64
|     +-- connection-ps?       yang:gauge64
|     +-- request-ps?          yang:gauge64
|     +-- partial-request-ps?  yang:gauge64
+-- total-attack-connection-port
|  +-- low-percentile-l* [protocol port]
|  |  +-- port                 inet:port-number
|  |  +-- protocol             uint8
|  |  +-- connection?          yang:gauge64
|  |  +-- embryonic?           yang:gauge64
|  |  +-- connection-ps?       yang:gauge64
|  |  +-- request-ps?          yang:gauge64
|  |  +-- partial-request-ps?  yang:gauge64
|  +-- mid-percentile-l* [protocol port]
|  |  +-- port                 inet:port-number
|  |  +-- protocol             uint8
|  |  +-- connection?          yang:gauge64
|  |  +-- embryonic?           yang:gauge64
|  |  +-- connection-ps?       yang:gauge64
|  |  +-- request-ps?          yang:gauge64
|  |  +-- partial-request-ps?  yang:gauge64
|  +-- high-percentile-l* [protocol port]
|  |  +-- port                 inet:port-number
|  |  +-- protocol             uint8
|  |  +-- connection?          yang:gauge64
|  |  +-- embryonic?           yang:gauge64
|  |  +-- connection-ps?       yang:gauge64
|  |  +-- request-ps?          yang:gauge64
```

```
                  |  |  +-- partial-request-ps?   yang:gauge64
                  |  +-- peak-l* [protocol port]
                  |  |  +-- port                inet:port-number
                  |  |  +-- protocol             uint8
                  |  |  +-- connection?          yang:gauge64
                  |  |  +-- embryonic?           yang:gauge64
                  |  |  +-- connection-ps?       yang:gauge64
                  |  |  +-- request-ps?          yang:gauge64
                  |  |  +-- partial-request-ps?   yang:gauge64
                  |  +-- current-l* [protocol port]
                  |     +-- port                inet:port-number
                  |     +-- protocol             uint8
                  |     +-- connection?          yang:gauge64
                  |     +-- embryonic?           yang:gauge64
                  |     +-- connection-ps?       yang:gauge64
                  |     +-- request-ps?          yang:gauge64
                  |     +-- partial-request-ps?   yang:gauge64
                  +-- attack-detail* [vendor-id attack-id]
                     ...
```

                Figure 28: Total Attack Connections Tree Structure

7.1.5. Attack Details

   This attribute (Figure 29) is used to signal a set of details
   characterizing an attack. The following subattributes describing the
   ongoing attack can be signal as attack details.

   vendor-id: Vendor ID is a security vendor's Enterprise Number as
      registered with IANA [Enterprise-Numbers]. It is a four-byte
      integer value.

   attack-id: Unique identifier assigned for the attack.

   attack-description: Textual representation of the attack
      description. Natural Language Processing techniques (e.g., word
      embedding) can possibly be used to map the attack description to
      an attack type. Textual representation of attack solves two
      problems: (a) avoids the need to create mapping tables manually
      between vendors and (b) avoids the need to standardize attack
      types which keep evolving.

   attack-severity: Attack severity level. This attribute takes one of
      the values defined in Section 3.12.2 of [RFC7970].

   start-time: The time the attack started. The attack's start time is
      expressed in seconds relative to 1970-01-01T00:00Z in UTC time

(Section 3.4.2 of [RFC8949]). The CBOR encoding is modified so
      that the leading tag 1 (epoch-based date/time) MUST be omitted.

   end-time: The time the attack ended. The attack end time is
      expressed in seconds relative to 1970-01-01T00:00Z in UTC time
      (Section 3.4.2 of [RFC8949]). The CBOR encoding is modified so
      that the leading tag 1 (epoch-based date/time) MUST be omitted.

   source-count: A count of sources involved in the attack targeting
      the victim.

   top-talker: A list of top talkers among attack sources. The top
      talkers are represented using the 'source-prefix'.

      'spoofed-status' indicates whether a top talker is a spoofed IP
      address (e.g., reflection attacks) or not.

      If the target is subjected to a bandwidth consuming attack, the
      attack traffic from each of the top talkers is included ('total-
      attack-traffic', Section 7.1.3).

      If the target is subjected to a resource consuming DDoS attack,
      the same attributes defined in Section 7.1.4 are applicable for
      representing the attack per talker.

```
    +--:(telemetry)
       +-- pre-or-ongoing-mitigation* []
          +-- (direction)?
          |  +--:(server-to-client-only)
          |     +-- tmid?                    uint32
          +-- target
          |  ...
          +-- total-traffic* [unit]
          |  ...
          +-- total-traffic-protocol* [unit protocol]
          |  ...
          +-- total-traffic-port* [unit port]
          |  ...
          +-- total-attack-traffic* [unit]
          |  ...
          +-- total-attack-traffic-protocol* [unit protocol]
          |  ...
          +-- total-attack-traffic-port* [unit port]
          |  ...
          +-- total-attack-connection
          |  ...
          +-- total-attack-connection-port
          |  ...
```

```
               +-- attack-detail* [vendor-id attack-id]
                  +-- vendor-id              uint32
                  +-- attack-id              uint32
                  +-- attack-description?    string
                  +-- attack-severity?       attack-severity
                  +-- start-time?            uint64
                  +-- end-time?              uint64
                  +-- source-count
                  |  +-- low-percentile-g?    yang:gauge64
                  |  +-- mid-percentile-g?    yang:gauge64
                  |  +-- high-percentile-g?   yang:gauge64
                  |  +-- peak-g?              yang:gauge64
                  |  +-- current-g?           yang:gauge64
                  +-- top-talker
                     +-- talker* [source-prefix]
                        +-- spoofed-status?            boolean
                        +-- source-prefix              inet:ip-prefix
                        +-- source-port-range* [lower-port]
                        |  +-- lower-port    inet:port-number
                        |  +-- upper-port?   inet:port-number
                        +-- source-icmp-type-range* [lower-type]
                        |  +-- lower-type    uint8
                        |  +-- upper-type?   uint8
                        +-- total-attack-traffic* [unit]
                        |  +-- unit              unit
                        |  +-- low-percentile-g?    yang:gauge64
                        |  +-- mid-percentile-g?    yang:gauge64
                        |  +-- high-percentile-g?   yang:gauge64
                        |  +-- peak-g?              yang:gauge64
                        |  +-- current-g?           yang:gauge64
                        +-- total-attack-connection
                           +-- low-percentile-l* [protocol]
                           |  +-- protocol            uint8
                           |  +-- connection?          yang:gauge64
                           |  +-- embryonic?           yang:gauge64
                           |  +-- connection-ps?       yang:gauge64
                           |  +-- request-ps?          yang:gauge64
                           |  +-- partial-request-ps?  yang:gauge64
                           +-- mid-percentile-l* [protocol]
                           |  +-- protocol            uint8
                           |  +-- connection?          yang:gauge64
                           |  +-- embryonic?           yang:gauge64
                           |  +-- connection-ps?       yang:gauge64
                           |  +-- request-ps?          yang:gauge64
                           |  +-- partial-request-ps?  yang:gauge64
                           +-- high-percentile-l* [protocol]
                           |  +-- protocol            uint8
                           |  +-- connection?          yang:gauge64
```

```
                        |   +-- embryonic?            yang:gauge64
                        |   +-- connection-ps?        yang:gauge64
                        |   +-- request-ps?           yang:gauge64
                        |   +-- partial-request-ps?   yang:gauge64
                        +-- peak-l* [protocol]
                        |   +-- protocol              uint8
                        |   +-- connection?           yang:gauge64
                        |   +-- embryonic?            yang:gauge64
                        |   +-- connection-ps?        yang:gauge64
                        |   +-- request-ps?           yang:gauge64
                        |   +-- partial-request-ps?   yang:gauge64
                        +-- current-l* [protocol]
                            +-- protocol              uint8
                            +-- connection?           yang:gauge64
                            +-- embryonic?            yang:gauge64
                            +-- connection-ps?        yang:gauge64
                            +-- request-ps?           yang:gauge64
                            +-- partial-request-ps?   yang:gauge64
```

                   Figure 29: Attack Detail Tree Structure

   In order to optimize the size of telemetry data conveyed over the
   DOTS signal channel, DOTS agents MAY use the DOTS data channel
   [RFC8783] to exchange vendor specific attack mapping details (that
   is, {vendor identifier, attack identifier} ==> attack description).
   As such, DOTS agents do not have to convey systematically an attack
   description in their telemetry messages over the DOTS signal channel.

   Multiple mappings for different vendor identifiers may be used; the
   DOTS agent transmitting telemetry information can elect to use one or
   more vendor mappings even in the same telemetry message.

      Note: It is possible that a DOTS server is making use of multiple
      DOTS mitigators; each from a different vendor. How telemetry
      information and vendor mappings are exchanged between DOTS servers
      and DOTS mitigators is outside the scope of this document.

   DOTS clients and servers may be provided with mappings from different
   vendors and so have their own different sets of vendor attack
   mappings. A DOTS agent MUST accept receipt of telemetry data with a
   vendor identifier that is different to the one it uses to transmit
   telemetry data. Furthermore, it is possible that the DOTS client and
   DOTS server are provided by the same vendor, but the vendor mapping
   tables are at different revisions. The DOTS client SHOULD transmit
   telemetry information using the vendor mapping(s) that it provided to
   the DOTS server and the DOTS server SHOULD use the vendor mappings(s)
   provided to the DOTS client when transmitting telemetry data to peer
   DOTS agent.

The "ietf-dots-mapping" YANG module defined in Section 10.2 augments
the "ietf-dots-data-channel" [RFC8783]. The tree structure of the
"ietf-dots-mapping" module is shown in Figure 30.

```
module: ietf-dots-mapping
  augment /data-channel:dots-data/data-channel:dots-client:
    +--rw vendor-mapping {dots-telemetry}?
       +--rw vendor* [vendor-id]
          +--rw vendor-id         uint32
          +--rw vendor-name?      string
          +--rw last-updated      uint64
          +--rw attack-mapping* [attack-id]
             +--rw attack-id           uint32
             +--rw attack-description     string
  augment /data-channel:dots-data/data-channel:capabilities:
    +--ro vendor-mapping-enabled?   boolean {dots-telemetry}?
  augment /data-channel:dots-data:
    +--ro vendor-mapping {dots-telemetry}?
       +--ro vendor* [vendor-id]
          +--ro vendor-id         uint32
          +--ro vendor-name?      string
          +--ro last-updated      uint64
          +--ro attack-mapping* [attack-id]
             +--ro attack-id           uint32
             +--ro attack-description     string
```

Figure 30: Vendor Attack Mapping Tree Structure

A DOTS client sends a GET request to retrieve the capabilities
supported by a DOTS server as per Section 7.1 of [RFC8783]. This
request is meant to assess whether the capability of sharing vendor
attack mapping details is supported by the server (i.e., check the
value of 'vendor-mapping-enabled').

If 'vendor-mapping-enabled' is set to 'true', A DOTS client MAY send
a GET request to retrieve the DOTS server's vendor attack mapping
details. An example of such GET request is shown in Figure 31.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /ietf-dots-mapping:vendor-mapping HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

Figure 31: GET to Retrieve the Vendor Attack Mappings of a DOTS
                            Server

A DOTS client MAY retrieve only the list of vendors supported by the
DOTS server. It does so by setting the "depth" parameter

(Section 4.8.2 of [RFC8040]) to "3" in the GET request as shown in
Figure 32. An example of a response body received from the DOTS
server as a response to such request is illustrated in Figure 33.

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /ietf-dots-mapping:vendor-mapping?depth=3 HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

  Figure 32: GET to Retrieve the Vendors List used by a DOTS Server

```
{
  "ietf-dots-mapping:vendor-mapping": {
    "vendor": [
      {
        "vendor-id": 1234,
        "vendor-name": "mitigator-s",
        "last-updated": "1576856561",
        "attack-mapping": []
      }
    ]
  }
}
```

   Figure 33: Response to a GET to Retrieve the Vendors List used by a
                             DOTS Server

The DOTS client reiterates the above procedure regularly (e.g., once
a week) to update the DOTS server's vendor attack mapping details.

If the DOTS client concludes that the DOTS server does not have any
reference to the specific vendor attack mapping details, the DOTS
client uses a POST request to install its vendor attack mapping
details. An example of such POST request is depicted in Figure 34.

```
POST /restconf/data/ietf-dots-data-channel:dots-data\
     /dots-client=dz6pHjaADkaFTbjr0JGBpw HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-dots-mapping:vendor-mapping": {
    "vendor": [
      {
        "vendor-id": 345,
        "vendor-name": "mitigator-c",
        "last-updated": "1576812345",
        "attack-mapping": [
          {
            "attack-id": 1,
            "attack-description":
                "Include a description of this attack"
          },
          {
            "attack-id": 2,
            "attack-description":
                "Again, include a description of the attack"
          }
        ]
      }
    ]
  }
}
```

          Figure 34: POST to Install Vendor Attack Mapping Details

   The DOTS server indicates the result of processing the POST request
   using the status-line. Concretely, "201 Created" status-line MUST be
   returned in the response if the DOTS server has accepted the vendor
   attack mapping details. If the request is missing a mandatory
   attribute or contains an invalid or unknown parameter, "400 Bad
   Request" status-line MUST be returned by the DOTS server in the
   response. The error-tag is set to "missing-attribute", "invalid-
   value", or "unknown-element" as a function of the encountered error.

   If the request is received via a server-domain DOTS gateway, but the
   DOTS server does not maintain a 'cdid' for this 'cuid' while a 'cdid'
   is expected to be supplied, the DOTS server MUST reply with "403
   Forbidden" status-line and the error-tag "access-denied". Upon
   receipt of this message, the DOTS client MUST register (Section 5.1
   of [RFC8783]).

The DOTS client uses the PUT request to modify its vendor attack
mapping details maintained by the DOTS server (e.g., add a new
mapping).

A DOTS client uses a GET request to retrieve its vendor attack
mapping details as maintained by the DOTS server (Figure 35).

```
GET /restconf/data/ietf-dots-data-channel:dots-data\
    /dots-client=dz6pHjaADkaFTbjr0JGBpw\
    /ietf-dots-mapping:vendor-mapping?\
    content=all HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

 Figure 35: GET to Retrieve Installed Vendor Attack Mapping Details

When conveying attack details in DOTS telemetry messages (Sections
7.2, 7.3, and 8), DOTS agents MUST NOT include 'attack-description'
attribute except if the corresponding attack mapping details were not
shared with the peer DOTS agent.

7.2. From DOTS Clients to DOTS Servers

DOTS clients uses PUT request to signal pre-or-ongoing-mitigation
telemetry to DOTS servers. An example of such request is shown in
Figure 36.

```
   Header: PUT (Code=0.03)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "tm"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "tmid=123"
   Content-Format: "application/dots+cbor"

   {
     "ietf-dots-telemetry:telemetry": {
       "pre-or-ongoing-mitigation": [
         {
           "target": {
             "target-prefix": [
               "2001:db8::1/128"
             ]
           },
           "total-attack-traffic-protocol": [
             {
               "protocol": 17,
               "unit": "megabit-ps",
               "mid-percentile-g": "900"
             }
           ],
           "attack-detail": [
             {
               "vendor-id": 1234,
               "attack-id": 77,
               "start-time": "1957811234",
               "attack-severity": "high"
             }
           ]
         }
       ]
     }
   }
```

         Figure 36: PUT to Send Pre-or-Ongoing-Mitigation Telemetry

   'cuid' is a mandatory Uri-Path parameter for PUT requests.

   The following additional Uri-Path parameter is defined:

   tmid: Telemetry Identifier is an identifier for the DOTS pre-or-
          ongoing-mitigation telemetry data represented as an integer.
          This identifier MUST be generated by DOTS clients. 'tmid' values
          MUST increase monotonically (when a new PUT is generated by a
          DOTS client to convey pre-or-ongoing-mitigation telemetry).

The procedure specified in Section 4.4.1 of
[I-D.ietf-dots-rfc8782-bis] MUST be followed for 'tmid'
rollover.

This is a mandatory attribute. 'tmid' MUST follow 'cuid'.

'cuid' and 'tmid' MUST NOT appear in the PUT request message body.

At least 'target' attribute and another pre-or-ongoing-mitigation
attributes (Section 7.1) MUST be present in the PUT request. If only
the 'target' attribute is present, this request is handled as per
Section 7.3.

The relative order of two PUT requests carrying DOTS pre-or-ongoing-
mitigation telemetry from a DOTS client is determined by comparing
their respective 'tmid' values. If such two requests have
overlapping 'target', the PUT request with higher numeric 'tmid'
value will override the request with a lower numeric 'tmid' value.
The overlapped lower numeric 'tmid' MUST be automatically deleted and
no longer be available.

The DOTS server indicates the result of processing a PUT request
using CoAP Response Codes. In particular, the 2.04 (Changed)
Response Code is returned if the DOTS server has accepted the pre-or-
ongoing-mitigation telemetry. The 5.03 (Service Unavailable)
Response Code is returned if the DOTS server has erred. 5.03 uses
Max-Age Option to indicate the number of seconds after which to
retry.

How long a DOTS server maintains a 'tmid' as active or logs the
enclosed telemetry information is implementation specific. Note that
if a 'tmid' is still active, then logging details are updated by the
DOTS server as a function of the updates received from the peer DOTS
client.

A DOTS client that lost the state of its active 'tmids' or has to set
'tmid' back to zero (e.g., crash or restart) MUST send a GET request
to the DOTS server to retrieve the list of active 'tmid'. The DOTS
client may then delete 'tmids' that should not be active anymore
(Figure 37). Sending a DELETE with no 'tmid' indicates that all
'tmids' must be deactivated (Figure 38).

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=123"
```

Figure 37: Delete a Pre-or-Ongoing-Mitigation Telemetry

```
Header: DELETE (Code=0.04)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
```

Figure 38: Delete All Pre-or-Ongoing-Mitigation Telemetry

7.3. From DOTS Servers to DOTS Clients

The pre-or-ongoing-mitigation (attack details, in particular) can
also be signaled from DOTS servers to DOTS clients. For example, the
DOTS server co-located with a DDoS detector collects monitoring
information from the target network, identifies DDoS attack using
statistical analysis or deep learning techniques, and signals the
attack details to the DOTS client.

The DOTS client can use the attack details to decide whether to
trigger a DOTS mitigation request or not. Furthermore, the security
operation personnel at the DOTS client domain can use the attack
details to determine the protection strategy and select the
appropriate DOTS server for mitigating the attack.

In order to receive pre-or-ongoing-mitigation telemetry notifications
from a DOTS server, a DOTS client MUST send a PUT (followed by a GET)
with the target filter. An example of such PUT request is shown in
Figure 39. In order to avoid maintaining a long list of such
requests, it is RECOMMENDED that DOTS clients include all targets in
the same request. DOTS servers may be instructed to restrict the
number of pre-or-ongoing-mitigation requests per DOTS client domain.
This request MUST be maintained active by the DOTS server until a
delete request is received from the same DOTS client to clear this
pre-or-ongoing-mitigation telemetry.

The relative order of two PUT requests carrying DOTS pre-or-ongoing-
mitigation telemetry from a DOTS client is determined by comparing
their respective 'tmid' values. If such two requests have
overlapping 'target', the PUT request with higher numeric 'tmid'
value will override the request with a lower numeric 'tmid' value.

The overlapped lower numeric 'tmid' MUST be automatically deleted and
no longer be available.

```
Header: PUT (Code=0.03)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=567"
Content-Format: "application/dots+cbor"

{
  "ietf-dots-telemetry:telemetry": {
    "pre-or-ongoing-mitigation": [
      {
        "target": {
          "target-prefix": [
            "2001:db8::/32"
          ]
        }
      }
    ]
  }
}
```

        Figure 39: PUT to Request Pre-or-Ongoing-Mitigation Telemetry

DOTS clients of the same domain can request to receive pre-or-
ongoing-mitigation telemetry bound to the same target.

The DOTS client conveys the Observe Option set to '0' in the GET
request to receive asynchronous notifications carrying pre-or-
ongoing-mitigation telemetry data from the DOTS server. The GET
request specifies a 'tmid' (Figure 40) or not (Figure 41).

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "tmid=567"
Observe: 0
```

 Figure 40: GET to Subscribe to Telemetry Asynchronous Notifications
                       for a Specific 'tmid'

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Observe: 0
```

Figure 41: GET to Subscribe to Telemetry Asynchronous Notifications
for All 'tmids'

The DOTS client can filter out the asynchronous notifications from
the DOTS server by indicating one or more Uri-Query options in its
GET request. A Uri-Query option can include the following
parameters: 'target-prefix', 'target-port', 'target-protocol',
'target-fqdn', 'target-uri', 'alias-name', 'mid', and 'c' (content)
(Section 4.2.4). Furthermore:

   If more than one Uri-Query option is included in a request, these
   options are interpreted in the same way as when multiple target
   attributes are included in a message body.

   If multiple values of a query parameter are to be included in a
   request, these values MUST be included in the same Uri-Query
   option and separated by a "," character without any spaces.

   Range values (i.e., contiguous inclusive block) can be included
   for 'target-port', 'target-protocol', and 'mid' parameters by
   indicating two bound values separated by a "-" character.

   Wildcard names (i.e., a name with the leftmost label is the "*"
   character) can be included in 'target-fqdn' or 'target-uri'
   parameters. DOTS clients MUST NOT include a name in which the "*"
   character is included in a label other than the leftmost label.
   "*.example.com" is an example of a valid wildcard name that can be
   included as a value of the 'target-fqdn' parameter in an Uri-Query
   option.

DOTS clients may also filter out the asynchronous notifications from
the DOTS server by indicating a specific source information. To that
aim, a DOTS client may include 'source-prefix', 'source-port', or
'source-icmp-type' in a Uri-Query option. The same considerations
(ranges, multiple values) specified for target attributes apply for
source attributes. Special care SHOULD be taken when using these
filters as some attacks may be hidden to the requesting DOTS client
(e.g., the attack changes its source information).

Requests with invalid query types (e.g., not supported, malformed) by
the DOTS server MUST be rejected by DOTS servers with a 4.00 (Bad
Request).

An example of request to subscribe to asynchronous UDP telemetry
notifications is shown in Figure 42. This filter will be applied for
all 'tmids'.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "tm"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Query: "target-protocol=17"
Observe: 0
```

Figure 42: GET Request to Receive Telemetry Asynchronous
Notifications Filtered using Uri-Query

The DOTS server will send asynchronous notifications to the DOTS
client when an attack event is detected following similar
considerations as in Section 4.4.2.1 of [I-D.ietf-dots-rfc8782-bis].
An example of a pre-or-ongoing-mitigation telemetry notification is
shown in Figure 43.

```
   {
     "ietf-dots-telemetry:telemetry": {
       "pre-or-ongoing-mitigation": [
         {
           "tmid": 567,
           "target": {
             "target-prefix": [
               "2001:db8::1/128"
             ]
           },
           "target-protocol": [
             17
           ],
           "total-attack-traffic": [
             {
               "unit": "megabit-ps",
               "mid-percentile-g": "900"
             }
           ],
           "attack-detail": [
             {
               "vendor-id": 1234,
               "attack-id": 77,
               "start-time": "1957818434",
               "attack-severity": "high"
             }
           ]
         }
       ]
     }
   }
```

         Figure 43: Message Body of a Pre-or-Ongoing-Mitigation Telemetry
                         Notification from the DOTS Server

   A DOTS server sends the aggregate data for a target using 'total-
   attack-traffic' attribute. The aggregate assumes that Uri-Query
   filters are applied on the target. The DOTS server MAY include more
   granular data when needed (that is, 'total-attack-traffic-protocol'
   and 'total-attack-traffic-port'). If a port filter (or protocol
   filter) is included in a request, 'total-attack-traffic-protocol' (or
   'total-attack-traffic-port') conveys the data with the port (or
   protocol) filter applied.

   A DOTS server may aggregate pre-or-ongoing-mitigation data (e.g.,
   'top-talker') for all targets of a domain, or when justified, send
   specific information (e.g., 'top-talker') per individual targets.

The DOTS client may log pre-or-ongoing-mitigation telemetry data with
an alert sent to an administrator or a network controller. The DOTS
client may send a mitigation request if the attack cannot be handled
locally.

A DOTS client that is not interested to receive pre-or-ongoing-
mitigation telemetry data for a target MUST send a delete request
similar to the one depicted in Figure 37.

8. DOTS Telemetry Mitigation Status Update

8.1. DOTS Clients to Servers Mitigation Efficacy DOTS Telemetry Attributes

The mitigation efficacy telemetry attributes can be signaled from
DOTS clients to DOTS servers as part of the periodic mitigation
efficacy updates to the server (Section 4.4.3 of
[I-D.ietf-dots-rfc8782-bis]).

Total Attack Traffic: The overall attack traffic as observed from
   the DOTS client perspective during an active mitigation. See
   Figure 27.

Attack Details: The overall attack details as observed from the
   DOTS client perspective during an active mitigation. See
   Section 7.1.5.

The "ietf-dots-telemetry" YANG module (Section 10.1) augments the
'mitigation-scope' message type defined in "ietf-dots-signal"
[I-D.ietf-dots-rfc8782-bis] so that these attributes can be signalled
by a DOTS client in a mitigation efficacy update (Figure 44).

```
  augment-structure /dots-signal:dots-signal/dots-signal:message-type
                /dots-signal:mitigation-scope/dots-signal:scope:
    +-- total-attack-traffic* [unit]
    |  +-- unit              unit
    |  +-- low-percentile-g?    yang:gauge64
    |  +-- mid-percentile-g?    yang:gauge64
    |  +-- high-percentile-g?   yang:gauge64
    |  +-- peak-g?           yang:gauge64
    |  +-- current-g?        yang:gauge64
    +-- attack-detail* [vendor-id attack-id]
        +-- vendor-id          uint32
        +-- attack-id          uint32
        +-- attack-description?   string
        +-- attack-severity?    attack-severity
        +-- start-time?        uint64
        +-- end-time?          uint64
```

```
        +-- source-count
        |  +-- low-percentile-g?     yang:gauge64
        |  +-- mid-percentile-g?     yang:gauge64
        |  +-- high-percentile-g?    yang:gauge64
        |  +-- peak-g?               yang:gauge64
        |  +-- current-g?              yang:gauge64
        +-- top-talker
           +-- talker* [source-prefix]
              +-- spoofed-status?          boolean
              +-- source-prefix            inet:ip-prefix
              +-- source-port-range* [lower-port]
              |  +-- lower-port     inet:port-number
              |  +-- upper-port?    inet:port-number
              +-- source-icmp-type-range* [lower-type]
              |  +-- lower-type     uint8
              |  +-- upper-type?    uint8
              +-- total-attack-traffic* [unit]
              |  +-- unit               unit
              |  +-- low-percentile-g?    yang:gauge64
              |  +-- mid-percentile-g?    yang:gauge64
              |  +-- high-percentile-g?   yang:gauge64
              |  +-- peak-g?              yang:gauge64
              |  +-- current-g?             yang:gauge64
              +-- total-attack-connection
                 +-- low-percentile-c
                 |  +-- connection?        yang:gauge64
                 |  +-- embryonic?         yang:gauge64
                 |  +-- connection-ps?     yang:gauge64
                 |  +-- request-ps?        yang:gauge64
                 |  +-- partial-request-ps?   yang:gauge64
                 +-- mid-percentile-c
                 |  ...
                 +-- high-percentile-c
                 |  ...
                 +-- peak-c
                 |  ...
                 +-- current-c
                    ...
```

Figure 44: Telemetry Efficacy Update Tree Structure

In order to signal telemetry data in a mitigation efficacy update, it
is RECOMMENDED that the DOTS client has already established a DOTS
telemetry setup session with the server in 'idle' time.

An example of an efficacy update with telemetry attributes is
depicted in Figure 45.

```
   Header: PUT (Code=0.03)
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "mitigate"
   Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
   Uri-Path: "mid=123"
   If-Match:
   Content-Format: "application/dots+cbor"

   {
     "ietf-dots-signal-channel:mitigation-scope": {
       "scope": [
         {
           "alias-name": [
             "https1",
             "https2"
           ],
           "attack-status": "under-attack",
           "ietf-dots-telemetry:total-attack-traffic": [
             {
               "unit": "megabit-ps",
               "mid-percentile-g": "900"
             }
           ]
         }
       ]
     }
   }
```

       Figure 45: An Example of Mitigation Efficacy Update with Telemetry
                                   Attributes

8.2. DOTS Servers to Clients Mitigation Status DOTS Telemetry
       Attributes

   The mitigation status telemetry attributes can be signaled from the
   DOTS server to the DOTS client as part of the periodic mitigation
   status update (Section 4.4.2.2 of [I-D.ietf-dots-rfc8782-bis]). In
   particular, DOTS clients can receive asynchronous notifications of
   the attack details from DOTS servers using the Observe option defined
   in [RFC7641].

   In order to make use of this feature, DOTS clients MUST establish a
   telemetry setup session with the DOTS server in 'idle' time and MUST
   set the 'server-originated-telemetry' attribute to 'true'.

DOTS servers MUST NOT include telemetry attributes in mitigation
status updates sent to DOTS clients for which 'server-originated-
telemetry' attribute is set to 'false'.

As defined in [RFC8612], the actual mitigation activities can include
several countermeasure mechanisms. The DOTS server signals the
current operational status of relevant countermeasures. A list of
attacks detected by each countermeasure MAY also be included. The
same attributes defined in Section 7.1.5 are applicable for
describing the attacks detected and mitigated at the DOTS server
domain.

The "ietf-dots-telemetry" YANG module (Section 10.1) augments the
'mitigation-scope' message type defined in "ietf-dots-signal"
[I-D.ietf-dots-rfc8782-bis] with telemetry data as depicted in the
following tree structure:

```
  augment-structure /dots-signal:dots-signal/dots-signal:message-type
                    /dots-signal:mitigation-scope/dots-signal:scope:
    +-- (direction)?
    |  +--:(server-to-client-only)
    |     +-- total-traffic* [unit]
    |     |  +-- unit                  unit
    |     |  +-- low-percentile-g?     yang:gauge64
    |     |  +-- mid-percentile-g?     yang:gauge64
    |     |  +-- high-percentile-g?    yang:gauge64
    |     |  +-- peak-g?               yang:gauge64
    |     |  +-- current-g?            yang:gauge64
    |     +-- total-attack-connection
    |        +-- low-percentile-c
    |        |  +-- connection?        yang:gauge64
    |        |  +-- embryonic?         yang:gauge64
    |        |  +-- connection-ps?     yang:gauge64
    |        |  +-- request-ps?        yang:gauge64
    |        |  +-- partial-request-ps?   yang:gauge64
    |        +-- mid-percentile-c
    |        |  ...
    |        +-- high-percentile-c
    |        |  ...
    |        +-- peak-c
    |        |  ...
    |        +-- current-c
    |           ...
    +-- total-attack-traffic* [unit]
    |  +-- unit                  unit
    |  +-- low-percentile-g?     yang:gauge64
    |  +-- mid-percentile-g?     yang:gauge64
    |  +-- high-percentile-g?    yang:gauge64
```

```
       |  +-- peak-g?              yang:gauge64
       |  +-- current-g?           yang:gauge64
      +-- attack-detail* [vendor-id attack-id]
         +-- vendor-id             uint32
         +-- attack-id             uint32
         +-- attack-description?   string
         +-- attack-severity?      attack-severity
         +-- start-time?           uint64
         +-- end-time?             uint64
         +-- source-count
         |  +-- low-percentile-g?    yang:gauge64
         |  +-- mid-percentile-g?    yang:gauge64
         |  +-- high-percentile-g?   yang:gauge64
         |  +-- peak-g?              yang:gauge64
         |  +-- current-g?           yang:gauge64
         +-- top-talker
            +-- talker* [source-prefix]
               +-- spoofed-status?          boolean
               +-- source-prefix            inet:ip-prefix
               +-- source-port-range* [lower-port]
               |  +-- lower-port    inet:port-number
               |  +-- upper-port?   inet:port-number
               +-- source-icmp-type-range* [lower-type]
               |  +-- lower-type    uint8
               |  +-- upper-type?   uint8
               +-- total-attack-traffic* [unit]
               |  +-- unit                  unit
               |  +-- low-percentile-g?    yang:gauge64
               |  +-- mid-percentile-g?    yang:gauge64
               |  +-- high-percentile-g?   yang:gauge64
               |  +-- peak-g?              yang:gauge64
               |  +-- current-g?           yang:gauge64
               +-- total-attack-connection
                  +-- low-percentile-c
                  |  +-- connection?          yang:gauge64
                  |  +-- embryonic?           yang:gauge64
                  |  +-- connection-ps?       yang:gauge64
                  |  +-- request-ps?          yang:gauge64
                  |  +-- partial-request-ps?  yang:gauge64
                  +-- mid-percentile-c
                  |  ...
                  +-- high-percentile-c
                  |  ...
                  +-- peak-c
                  |  ...
                  +-- current-c
                     ...
```

Figure 46 shows an example of an asynchronous notification of attack
mitigation status from the DOTS server. This notification signals
both the mid-percentile value of processed attack traffic and the
peak percentile value of unique sources involved in the attack.

```
{
  "ietf-dots-signal-channel:mitigation-scope": {
    "scope": [
      {
        "mid": 12332,
        "mitigation-start": "1507818434",
        "alias-name": [
          "https1",
          "https2"
        ],
        "lifetime": 1600,
        "status": "attack-successfully-mitigated",
        "bytes-dropped": "134334555",
        "bps-dropped": "43344",
        "pkts-dropped": "333334444",
        "pps-dropped": "432432",
        "ietf-dots-telemetry:total-attack-traffic": [
          {
            "unit": "megabit-ps",
            "mid-percentile-g": "900"
          }
        ],
        "ietf-dots-telemetry:attack-detail": [
          {
            "vendor-id": 1234,
            "attack-id": 77,
            "source-count": {
              "peak-g": "10000"
            }
          }
        ]
      }
    ]
  }
}
```

Figure 46: Response Body of a Mitigation Status With Telemetry
Attributes

DOTS clients can filter out the asynchronous notifications from the
DOTS server by indicating one or more Uri-Query options in its GET
request. A Uri-Query option can include the following parameters:
'target-prefix', 'target-port', 'target-protocol', 'target-fqdn',

'target-uri', 'alias-name', and 'c' (content) (Section 4.2.4). The
considerations discussed in Section 7.3 MUST be followed to include
multiple query values, ranges ('target-port', 'target-protocol'), and
wildcard name ('target-fqdn', 'target-uri').

An example of request to subscribe to asynchronous notifications
bound to the "http1" alias is shown in Figure 47.

```
Header: GET (Code=0.01)
Uri-Path: ".well-known"
Uri-Path: "dots"
Uri-Path: "mitigate"
Uri-Path: "cuid=dz6pHjaADkaFTbjr0JGBpw"
Uri-Path: "mid=12332"
Uri-Query: "target-alias=https1"
Observe: 0
```

Figure 47: GET Request to Receive Asynchronous Notifications Filtered
using Uri-Query

If the target query does not match the target of the enclosed 'mid'
as maintained by the DOTS server, the latter MUST respond with a 4.04
(Not Found) error Response Code. The DOTS server MUST NOT add a new
observe entry if this query overlaps with an existing one.

9. Error Handling

A list of common CoAP errors that are implemented by DOTS servers are
provided in Section 9 of [I-D.ietf-dots-rfc8782-bis]. The following
additional error cases apply for the telemetry extension:

o 4.00 (Bad Request) is returned by the DOTS server when the DOTS
   client has sent a request that violates the DOTS telemetry
   extension.

o 4.04 (Not Found) is returned by the DOTS server when the DOTS
   client is requesting a 'tsid' or 'tmid' that is not valid.

o 4.00 (Bad Request) is returned by the DOTS server when the DOTS
   client has sent a request with invalid query types (e.g., not
   supported, malformed).

o 4.04 (Not Found) is returned by the DOTS server when the DOTS
   client has sent a request with a target query that does not match
   the target of the enclosed 'mid' as maintained by the DOTS server.

10. YANG Modules

10.1. DOTS Signal Channel Telemetry YANG Module

   This module uses types defined in [RFC6991] and [RFC8345].

      Note to the RFC Editor: Please replace "RFC UUUU" with the RFC
      number to be assigned to [I-D.ietf-dots-rfc8782-bis].

```
<CODE BEGINS> file "ietf-dots-telemetry@2020-12-07.yang"
module ietf-dots-telemetry {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-telemetry";
  prefix dots-telemetry;

  import ietf-dots-signal-channel {
    prefix dots-signal;
    reference
      "RFC UUUU: Distributed Denial-of-Service Open Threat Signaling
                 (DOTS) Signal Channel Specification";
  }
  import ietf-dots-data-channel {
    prefix data-channel;
    reference
      "RFC 8783: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Data Channel Specification";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "Section 3 of RFC 6991";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "Section 4 of RFC 6991";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "Section 6.2 of RFC 8345: A YANG Data Model for Network
       Topologies";
  }
  import ietf-yang-structure-ext {
    prefix sx;
    reference
      "RFC 8791: YANG Data Structure Extensions";
  }
```

```
  organization
    "IETF DDoS Open Threat Signaling (DOTS) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/dots/>
     WG List:  <mailto:dots@ietf.org>

     Author:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>

     Author:   Konda, Tirumaleswar Reddy
               <mailto:TirumaleswarReddy_Konda@McAfee.com>";
  description
    "This module contains YANG definitions for the signaling
     of DOTS telemetry exchanged between a DOTS client and
     a DOTS server by means of the DOTS signal channel.

     Copyright (c) 2020 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2020-12-07 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Telemetry";
  }

  typedef attack-severity {
    type enumeration {
      enum none {
        value 1;
        description
          "No effect on the DOTS client domain.";
      }
      enum low {
        value 2;
        description
          "Minimal effect on the DOTS client domain.";
```

```
      }
      enum medium {
        value 3;
        description
          "A subset of DOTS client domain resources are
           out of service.";
      }
      enum high {
        value 4;
        description
          "The DOTS client domain is under extremly severe
           conditions.";
      }
      enum unknown {
        value 5;
        description
          "The impact of the attack is not known.";
      }
    }
    description
      "Enumeration for attack severity.";
    reference
      "RFC 7970: The Incident Object Description Exchange
                 Format Version 2";
  }

  typedef unit-class {
    type enumeration {
      enum packet-ps {
        value 1;
        description
          "Packets per second (pps).";
      }
      enum bit-ps {
        value 2;
        description
          "Bits per Second (bit/s).";
      }
      enum byte-ps {
        value 3;
        description
          "Bytes per second (Byte/s).";
      }
    }
    description
      "Enumeration to indicate which unit class is used.
       These classes are supported: pps, bit/s, and Byte/s.";
  }
```

```
   typedef unit {
     type enumeration {
       enum packet-ps {
         value 1;
         description
           "Packets per second (pps).";
       }
       enum bit-ps {
         value 2;
         description
           "Bits per Second (bps).";
       }
       enum byte-ps {
         value 3;
         description
           "Bytes per second (Bps).";
       }
       enum kilopacket-ps {
         value 4;
         description
           "Kilo packets per second (kpps).";
       }
       enum kilobit-ps {
         value 5;
         description
           "Kilobits per second (kbps).";
       }
       enum kilobyte-ps {
         value 6;
         description
           "Kilobytes per second (kBps).";
       }
       enum megapacket-ps {
         value 7;
         description
           "Mega packets per second (Mpps).";
       }
       enum megabit-ps {
         value 8;
         description
           "Megabits per second (Mbps).";
       }
       enum megabyte-ps {
         value 9;
         description
           "Megabytes per second (MBps).";
       }
       enum gigapacket-ps {
```

```
      value 10;
      description
        "Giga packets per second (Gpps).";
    }
    enum gigabit-ps {
      value 11;
      description
        "Gigabits per second (Gbps).";
    }
    enum gigabyte-ps {
      value 12;
      description
        "Gigabytes per second (GBps).";
    }
    enum terapacket-ps {
      value 13;
      description
        "Tera packets per second (Tpps).";
    }
    enum terabit-ps {
      value 14;
      description
        "Terabits per second (Tbps).";
    }
    enum terabyte-ps {
      value 15;
      description
        "Terabytes per second (TBps).";
    }
    enum petapacket-ps {
      value 16;
      description
        "Peta packets per second (Ppps).";
    }
    enum petabit-ps {
      value 17;
      description
        "Petabits per second (Pbps).";
    }
    enum petabyte-ps {
      value 18;
      description
        "Exabytes per second (PBps).";
    }
    enum exapacket-ps {
      value 19;
      description
        "Exa packets per second (Epps).";
```

```
      }
      enum exabit-ps {
        value 20;
        description
          "Exabits per second (Ebps).";
      }
      enum exabyte-ps {
        value 21;
        description
          "Exabytes per second (EBps).";
      }
      enum zettapacket-ps {
        value 22;
        description
          "Zetta packets per second (Zpps).";
      }
      enum zettabit-ps {
        value 23;
        description
          "Zettabits per second (Zbps).";
      }
      enum zettabyte-ps {
        value 24;
        description
          "Zettabytes per second (ZBps).";
      }
    }
    description
      "Enumeration to indicate which unit is used.
       Only one unit per unit class is used owing to
       unit auto-scaling.";
  }

  typedef interval {
    type enumeration {
      enum 5-minutes {
        value 1;
        description
          "5 minutes.";
      }
      enum 10-minutes {
        value 2;
        description
          "10 minutes.";
      }
      enum 30-minutes {
        value 3;
        description
```

```
          "30 minutes.";
      }
      enum hour {
        value 4;
        description
          "Hour.";
      }
      enum day {
        value 5;
        description
          "Day.";
      }
      enum week {
        value 6;
        description
          "Week.";
      }
      enum month {
        value 7;
        description
          "Month.";
      }
    }
    description
      "Enumeration to indicate the overall measurement period.";
  }

  typedef sample {
    type enumeration {
      enum second {
        value 1;
        description
          "A one second measurement period.";
      }
      enum 5-seconds {
        value 2;
        description
          "5 seconds measurement period.";
      }
      enum 30-seconds {
        value 3;
        description
          "30 seconds measurement period.";
      }
      enum minute {
        value 4;
        description
          "One minute measurement period.";
```

```
      }
      enum 5-minutes {
        value 5;
        description
          "5 minutes measurement period.";
      }
      enum 10-minutes {
        value 6;
        description
          "10 minutes measurement period.";
      }
      enum 30-minutes {
        value 7;
        description
          "30 minutes measurement period.";
      }
      enum hour {
        value 8;
        description
          "One hour measurement period.";
      }
    }
    description
      "Enumeration to indicate the sampling period.";
  }

  typedef percentile {
    type decimal64 {
      fraction-digits 2;
    }
    description
      "The nth percentile of a set of data is the
       value at which n percent of the data is below it.";
  }

  typedef query-type {
    type enumeration {
      enum target-prefix {
        value 1;
        description
          "Query based on target prefix.";
      }
      enum target-port {
        value 2;
        description
          "Query based on target port number.";
      }
      enum target-protocol {
```

```
          value 3;
          description
            "Query based on target protocol.";
        }
        enum target-fqdn {
          value 4;
          description
            "Query based on target FQDN.";
        }
        enum target-uri {
          value 5;
          description
            "Query based on target URI.";
        }
        enum target-alias {
          value 6;
          description
            "Query based on target alias.";
        }
        enum mid {
          value 7;
          description
            "Query based on mitigation identifier (mid).";
        }
        enum source-prefix {
          value 8;
          description
            "Query based on source prefix.";
        }
        enum source-port {
          value 9;
          description
            "Query based on source port number.";
        }
        enum source-icmp-type {
          value 10;
          description
            "Query based on ICMP type";
        }
        enum content {
          value 11;
          description
            "Query based on 'c' Uri-Query option that is used
             to control the selection of configuration
              and non-configuration data nodes.";
          reference
            "Section 4.4.2 of RFC UUUU.";
        }
```

```
      }
      description
        "Enumeration support for query types that can be used
         in a GET request to filter out data. Requests with
         invalid query types (e.g., not supported, malformed)
         by the DOTS server are rejected by DOTS servers with
         a 4.00 (Bad Request).";
  }

  grouping telemetry-parameters {
    description
      "A grouping that includes a set of parameters that
       are used to compute telemetry data.

       The grouping indicates a measurement interval,
       a measurement sample period, and low/mid/high
       percentile values.";
    leaf measurement-interval {
      type interval;
      description
        "Defines the period on which percentiles are computed.";
    }
    leaf measurement-sample {
      type sample;
      description
        "Defines the time distribution for measuring
         values that are used to compute percentiles.

         The measurement sample value must be less than the
         measurement interval value.";
    }
    leaf low-percentile {
      type percentile;
      default "10.00";
      description
        "Low percentile. If set to '0', this means low-percentiles
         are disabled.";
    }
    leaf mid-percentile {
      type percentile;
      must '. >= ../low-percentile' {
        error-message
          "The mid-percentile must be greater than
           or equal to the low-percentile.";
      }
      default "50.00";
      description
        "Mid percentile. If set to the same value as low-percentiles,
```

```
          this means mid-percentiles are disabled.";
    }
    leaf high-percentile {
      type percentile;
      must '. >= ../mid-percentile' {
        error-message
          "The high-percentile must be greater than
           or equal to the mid-percentile.";
      }
      default "90.00";
      description
        "High percentile. If set to the same value as mid-percentiles,
         this means high-percentiles are disabled.";
    }
  }

  grouping percentile-and-peak {
    description
      "Generic grouping for percentile and peak values.";
    leaf low-percentile-g {
      type yang:gauge64;
      description
        "Low percentile value.";
    }
    leaf mid-percentile-g {
      type yang:gauge64;
      description
        "Mid percentile value.";
    }
    leaf high-percentile-g {
      type yang:gauge64;
      description
        "High percentile value.";
    }
    leaf peak-g {
      type yang:gauge64;
      description
        "Peak value.";
    }
  }

  grouping unit-config {
    description
      "Generic grouping for unit configuration.";
    list unit-config {
      key "unit";
      description
        "Controls which unit classes are allowed when sharing
```

```
             telemetry data.";
         leaf unit {
           type unit-class;
           description
             "Can be packet-ps, bit-ps, or byte-ps.";
         }
         leaf unit-status {
           type boolean;
           default true;
           description
             "Enable/disable the use of the measurement unit class.";
         }
       }
     }

     grouping traffic-unit {
       description
         "Grouping of traffic as a function of the measurement unit.";
       leaf unit {
         type unit;
         description
           "The traffic can be measured using unit classes: packet-ps,
            bit-ps, or byte-ps. DOTS agents auto-scale to the appropriate
            units (e.g., megabit-ps, kilobit-ps).";
       }
       uses percentile-and-peak;
     }

     grouping traffic-unit-all {
       description
         "Grouping of traffic as a function of the measurement unit,
          including current values.";
       uses traffic-unit;
       leaf current-g {
         type yang:gauge64;
         description
           "Current observed value.";
       }
     }

     grouping traffic-unit-protocol {
       description
         "Grouping of traffic of a given transport protocol as
          a function of the measurement unit.";
       leaf protocol {
         type uint8;
         description
           "The transport protocol.
```

```
          Values are taken from the IANA Protocol Numbers registry:
          <https://www.iana.org/assignments/protocol-numbers/>.

          For example, this parameter contains 6 for TCP,
          17 for UDP, 33 for DCCP, or 132 for SCTP.";
    }
    uses traffic-unit;
  }

  grouping traffic-unit-protocol-all {
    description
      "Grouping of traffic of a given transport protocol as,
       including current values.";
    uses traffic-unit-protocol;
    leaf current-g {
      type yang:gauge64;
      description
        "Current observed value.";
    }
  }

  grouping traffic-unit-port {
    description
      "Grouping of traffic bound to a port number as
       a function of the measurement unit.";
    leaf port {
      type inet:port-number;
      description
        "Port number used by a transport protocol.";
    }
    uses traffic-unit;
  }

  grouping traffic-unit-port-all {
    description
      "Grouping of traffic bound to a port number as
       a function of the measurement unit, including
       current values.";
    uses traffic-unit-port;
    leaf current-g {
      type yang:gauge64;
      description
        "Current observed value.";
    }
  }

  grouping total-connection-capacity {
    description
```

```
      "Total connections capacity. These data nodes are
       useful to detect resource consuming DDoS attacks.";
    leaf connection {
      type uint64;
      description
        "The maximum number of simultaneous connections that
         are allowed to the target server.";
    }
    leaf connection-client {
      type uint64;
      description
        "The maximum number of simultaneous connections that
         are allowed to the target server per client.";
    }
    leaf embryonic {
      type uint64;
      description
        "The maximum number of simultaneous embryonic connections
         that are allowed to the target server. The term 'embryonic
         connection' refers to a connection whose connection handshake
         is not finished. Embryonic connection is only possible in
         connection-oriented transport protocols like TCP or SCTP.";
    }
    leaf embryonic-client {
      type uint64;
      description
        "The maximum number of simultaneous embryonic connections
         that are allowed to the target server per client.";
    }
    leaf connection-ps {
      type uint64;
      description
        "The maximum number of connections allowed per second
         to the target server.";
    }
    leaf connection-client-ps {
      type uint64;
      description
        "The maximum number of connections allowed per second
         to the target server per client.";
    }
    leaf request-ps {
      type uint64;
      description
        "The maximum number of requests allowed per second
         to the target server.";
    }
    leaf request-client-ps {
```

```
        type uint64;
        description
          "The maximum number of requests allowed per second
           to the target server per client.";
      }
      leaf partial-request-ps {
        type uint64;
        description
          "The maximum number of partial requests allowed per
           second to the target server.";
      }
      leaf partial-request-client-ps {
        type uint64;
        description
          "The maximum number of partial requests allowed per
           second to the target server per client.";
      }
    }

    grouping total-connection-capacity-protocol {
      description
        "Total connections capacity per protocol. These data nodes are
         useful to detect resource consuming DDoS attacks.";
      leaf protocol {
        type uint8;
        description
          "The transport protocol.
           Values are taken from the IANA Protocol Numbers registry:
           <https://www.iana.org/assignments/protocol-numbers/>.";
      }
      uses total-connection-capacity;
    }

    grouping connection {
      description
        "A set of data nodes which represent the attack
         characteristics.";
      leaf connection {
        type yang:gauge64;
        description
          "The number of simultaneous attack connections to
           the target server.";
      }
      leaf embryonic {
        type yang:gauge64;
        description
          "The number of simultaneous embryonic connections to
           the target server.";
```

```
      }
      leaf connection-ps {
        type yang:gauge64;
        description
          "The number of attack connections per second to
           the target server.";
      }
      leaf request-ps {
        type yang:gauge64;
        description
          "The number of attack requests per second to
           the target server.";
      }
      leaf partial-request-ps {
        type yang:gauge64;
        description
          "The number of attack partial requests to
           the target server.";
      }
    }

    grouping connection-percentile-and-peak {
      description
        "Total attack connections. Low/mid/high percentile
         and peak values are included.";
      container low-percentile-c {
        description
          "Low percentile of attack connections.";
        uses connection;
      }
      container mid-percentile-c {
        description
          "Mid percentile of attack connections.";
        uses connection;
      }
      container high-percentile-c {
        description
          "High percentile of attack connections.";
        uses connection;
      }
      container peak-c {
        description
          "Peak attack connections.";
        uses connection;
      }
    }

    grouping connection-all {
```

```
      description
        "Total attack connections including current values.";
      uses connection-percentile-and-peak;
      container current-c {
        description
          "Current attack connections.";
        uses connection;
      }
    }

    grouping connection-protocol {
      description
        "Total attack connections.";
      leaf protocol {
        type uint8;
        description
          "The transport protocol.
           Values are taken from the IANA Protocol Numbers registry:
           <https://www.iana.org/assignments/protocol-numbers/>.";
      }
      uses connection;
    }

    grouping connection-port {
      description
        "Total attack connections per port number.";
      leaf port {
        type inet:port-number;
        description
          "Port number.";
      }
      uses connection-protocol;
    }

    grouping connection-protocol-percentile {
      description
        "Total attack connections per protocol.";
      list low-percentile-l {
        key "protocol";
        description
          "Low percentile of attack connections per protocol.";
        uses connection-protocol;
      }
      list mid-percentile-l {
        key "protocol";
        description
          "Mid percentile of attack connections per protocol.";
        uses connection-protocol;
```

```
      }
      list high-percentile-l {
        key "protocol";
        description
          "High percentile of attack connections per protocol.";
        uses connection-protocol;
      }
      list peak-l {
        key "protocol";
        description
          "Peak attack connections per protocol.";
        uses connection-protocol;
      }
    }

    grouping connection-protocol-all {
      description
        "Total attack connections per protocol, including current
         values.";
      uses connection-protocol-percentile;
      list current-l {
        key "protocol";
        description
          "Current attack connections per protocol.";
        uses connection-protocol;
      }
    }

    grouping connection-protocol-port-percentile {
      description
        "Total attack connections per port number experessed in
         low/mid/high percentile and peak values.";
      list low-percentile-l {
        key "protocol port";
        description
          "Low percentile of attack connections per port number.";
        uses connection-port;
      }
      list mid-percentile-l {
        key "protocol port";
        description
          "Mid percentile of attack connections per port number.";
        uses connection-port;
      }
      list high-percentile-l {
        key "protocol port";
        description
          "High percentile of attack connections per port number.";
```

```
      uses connection-port;
    }
    list peak-l {
      key "protocol port";
      description
        "Peak attack connections per port number.";
      uses connection-port;
    }
  }

  grouping connection-protocol-port-all {
    description
      "Total attack connections per port number, including current
       values.";
    uses connection-protocol-port-percentile;
    list current-l {
      key "protocol port";
      description
        "Current attack connections per port number.";
      uses connection-port;
    }
  }

  grouping attack-detail {
    description
      "Various details that describe the on-going
       attacks that need to be mitigated by the DOTS server.
       The attack details need to cover well-known and common attacks
       (such as a SYN Flood) along with new emerging or vendor-specific
       attacks.";
    leaf vendor-id {
      type uint32;
      description
        "Vendor ID is a security vendor's Enterprise Number.";
    }
    leaf attack-id {
      type uint32;
      description
        "Unique identifier assigned by the vendor for the attack.";
    }
    leaf attack-description {
      type string;
      description
        "Textual representation of attack description. Natural Language
         Processing techniques (e.g., word embedding) can possibly be
         used to map the attack description to an attack type.";
    }
    leaf attack-severity {
```

```
        type attack-severity;
        description
          "Severity level of an attack. How this level is determined
           is implementation-specific.";
      }
      leaf start-time {
        type uint64;
        description
          "The time the attack started. Start time is represented in
           seconds relative to 1970-01-01T00:00:00Z in UTC time.";
      }
      leaf end-time {
        type uint64;
        description
          "The time the attack ended. End time is represented in seconds
           relative to 1970-01-01T00:00:00Z in UTC time.";
      }
      container source-count {
        description
          "Indicates the count of unique sources involved
           in the attack.";
        uses percentile-and-peak;
        leaf current-g {
          type yang:gauge64;
          description
            "Current observed value.";
        }
      }
    }

    grouping top-talker-aggregate {
      description
        "An aggregate of top attack sources. This aggregate is
         typically used when included in a mitigation request.";
      list talker {
        key "source-prefix";
        description
          "IPv4 or IPv6 prefix identifying the attacker(s).";
        leaf spoofed-status {
          type boolean;
          description
            "Indicates whether this address is spoofed.";
        }
        leaf source-prefix {
          type inet:ip-prefix;
          description
            "IPv4 or IPv6 prefix identifying the attacker(s).";
        }
```

```
      list source-port-range {
        key "lower-port";
        description
          "Port range. When only lower-port is
           present, it represents a single port number.";
        leaf lower-port {
          type inet:port-number;
          description
            "Lower port number of the port range.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            error-message
              "The upper port number must be greater than
               or equal to lower port number.";
          }
          description
            "Upper port number of the port range.";
        }
      }
      list source-icmp-type-range {
        key "lower-type";
        description
          "ICMP type range. When only lower-type is
           present, it represents a single ICMP type.";
        leaf lower-type {
          type uint8;
          description
            "Lower ICMP type of the ICMP type range.";
        }
        leaf upper-type {
          type uint8;
          must '. >= ../lower-type' {
            error-message
              "The upper ICMP type must be greater than
               or equal to lower ICMP type.";
          }
          description
            "Upper type of the ICMP type range.";
        }
      }
      list total-attack-traffic {
        key "unit";
        description
          "Total attack traffic issued from this source.";
        uses traffic-unit-all;
      }
```

```
      container total-attack-connection {
        description
          "Total attack connections issued from this source.";
        uses connection-all;
      }
    }
  }

  grouping top-talker {
    description
      "Top attack sources.";
    list talker {
      key "source-prefix";
      description
        "IPv4 or IPv6 prefix identifying the attacker(s).";
      leaf spoofed-status {
        type boolean;
        description
          "Indicates whether this address is spoofed.";
      }
      leaf source-prefix {
        type inet:ip-prefix;
        description
          "IPv4 or IPv6 prefix identifying the attacker(s).";
      }
      list source-port-range {
        key "lower-port";
        description
          "Port range. When only lower-port is
           present, it represents a single port number.";
        leaf lower-port {
          type inet:port-number;
          description
            "Lower port number of the port range.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            error-message
              "The upper port number must be greater than
               or equal to lower port number.";
          }
          description
            "Upper port number of the port range.";
        }
      }
      list source-icmp-type-range {
        key "lower-type";
```

```
          description
            "ICMP type range. When only lower-type is
             present, it represents a single ICMP type.";
          leaf lower-type {
            type uint8;
            description
              "Lower ICMP type of the ICMP type range.";
          }
          leaf upper-type {
            type uint8;
            must '. >= ../lower-type' {
              error-message
                "The upper ICMP type must be greater than
                 or equal to lower ICMP type.";
            }
            description
              "Upper type of the ICMP type range.";
          }
        }
        list total-attack-traffic {
          key "unit";
          description
            "Total attack traffic issued from this source.";
          uses traffic-unit-all;
        }
        container total-attack-connection {
          description
            "Total attack connections issued from this source.";
          uses connection-protocol-all;
        }
      }
    }
  }

  grouping baseline {
    description
      "Grouping for the telemetry baseline.";
    uses data-channel:target;
    leaf-list alias-name {
      type string;
      description
        "An alias name that points to an IP resource.
         An IP resource can be be a router, a host,
         an IoT object, a server, etc.";
    }
    list total-traffic-normal {
      key "unit";
      description
        "Total traffic normal baselines.";
```

```
      uses traffic-unit;
    }
    list total-traffic-normal-per-protocol {
      key "unit protocol";
      description
        "Total traffic normal baselines per protocol.";
      uses traffic-unit-protocol;
    }
    list total-traffic-normal-per-port {
      key "unit port";
      description
        "Total traffic normal baselines per port number.";
      uses traffic-unit-port;
    }
    list total-connection-capacity {
      key "protocol";
      description
        "Total connection capacity.";
      uses total-connection-capacity-protocol;
    }
    list total-connection-capacity-per-port {
      key "protocol port";
      description
        "Total connection capacity per port number.";
      leaf port {
        type inet:port-number;
        description
          "The target port number.";
      }
      uses total-connection-capacity-protocol;
    }
  }

  grouping pre-or-ongoing-mitigation {
    description
      "Grouping for the telemetry data.";
    list total-traffic {
      key "unit";
      description
        "Total traffic.";
      uses traffic-unit-all;
    }
    list total-traffic-protocol {
      key "unit protocol";
      description
        "Total traffic per protocol.";
      uses traffic-unit-protocol-all;
    }
```

```
   list total-traffic-port {
     key "unit port";
     description
       "Total traffic per port number.";
     uses traffic-unit-port-all;
   }
   list total-attack-traffic {
     key "unit";
     description
       "Total attack traffic.";
     uses traffic-unit-protocol-all;
   }
   list total-attack-traffic-protocol {
     key "unit protocol";
     description
       "Total attack traffic per protocol.";
     uses traffic-unit-protocol-all;
   }
   list total-attack-traffic-port {
     key "unit port";
     description
       "Total attack traffic per port number.";
     uses traffic-unit-port-all;
   }
   container total-attack-connection {
     description
       "Total attack connections.";
     uses connection-protocol-all;
   }
   container total-attack-connection-port {
     description
       "Total attack connections.";
     uses connection-protocol-port-all;
   }
   list attack-detail {
     key "vendor-id attack-id";
     description
       "Provides a set of attack details.";
     uses attack-detail;
     container top-talker {
       description
         "Lists the top attack sources.";
       uses top-talker;
     }
   }
 }

 sx:augment-structure "/dots-signal:dots-signal"
```

```
                        + "/dots-signal:message-type"
                        + "/dots-signal:mitigation-scope"
                        + "/dots-signal:scope" {
      description
        "Extends mitigation scope with telemetry update data.";
      choice direction {
        description
          "Indicates the communication direction in which the
           data nodes can be included.";
        case server-to-client-only {
          description
            "These data nodes appear only in a mitigation message
             sent from the server to the client.";
          list total-traffic {
            key "unit";
            description
              "Total traffic.";
            uses traffic-unit-all;
          }
          container total-attack-connection {
            description
              "Total attack connections.";
            uses connection-all;
          }
        }
      }
      list total-attack-traffic {
        key "unit";
        description
          "Total attack traffic.";
        uses traffic-unit-all;
      }
      list attack-detail {
        key "vendor-id attack-id";
        description
          "Attack details";
        uses attack-detail;
        container top-talker {
          description
            "Top attack sources.";
          uses top-talker-aggregate;
        }
      }
    }
  }
  sx:structure dots-telemetry {
    description
      "Main structure for DOTS telemetry messages.";
    choice telemetry-message-type {
```

```
        description
          "Can be a telemetry-setup or telemetry data.";
        case telemetry-setup {
          description
            "Indicates the message is about telemetry.";
          choice direction {
            description
              "Indicates the communication direction in which the
               data nodes can be included.";
            case server-to-client-only {
              description
                "These data nodes appear only in a mitigation message
                 sent from the server to the client.";
              container max-config-values {
                description
                  "Maximum acceptable configuration values.";
                uses telemetry-parameters;
                leaf server-originated-telemetry {
                  type boolean;
                  default false;
                  description
                    "Indicates whether the DOTS server can be instructed
                     to send pre-or-ongoing-mitigation telemetry. If set
                     to FALSE or the data node is not present, this is
                     an indication that the server does not support this
                     capability.";
                }
                leaf telemetry-notify-interval {
                  type uint32 {
                    range "1 .. 3600";
                  }
                  units "seconds";
                  must ". >= ../../min-config-values"
                     + "/telemetry-notify-interval" {
                    error-message
                      "The value must be greater than or equal
                       to the telemetry-notify-interval in the
                       min-config-values";
                  }
                  description
                    "Minimum number of seconds between successive
                     telemetry notifications.";
                }
              }
              container min-config-values {
                description
                  "Minimum acceptable configuration values.";
                uses telemetry-parameters;
```

```
        leaf telemetry-notify-interval {
          type uint32 {
            range "1 .. 3600";
          }
          units "seconds";
          description
            "Minimum number of seconds between successive
             telemetry notifications.";
        }
      }
      container supported-unit-classes {
        description
          "Supported unit classes and default activation
           status.";
        uses unit-config;
      }
      leaf-list query-type {
        type query-type;
        description
          "Indicates which query types are supported by
           the server. If the server does not announce
           the query types it supports, the client will
           be unable to use any of the potential
           query-type to reduce the returned data
           content from the server.";
      }
    }
  }
  list telemetry {
    description
      "The telemetry data per DOTS client.  The keys
       of the list are 'cuid' and 'tsid', but these keys are not
       represented here because these keys are conveyed as
       mandatory Uri-Paths in requests.  Omitting keys
       is compliant with RFC8791.";
    choice direction {
      description
        "Indicates the communication direction in which the
         data nodes can be included.";
      case server-to-client-only {
        description
          "These data nodes appear only in a mitigation message
           sent from the server to the client.";
        leaf tsid {
          type uint32;
          description
            "An identifier for the DOTS telemetry setup
             data.";
```

```
                }
              }
            }
          choice setup-type {
            description
              "Can be a mitigation configuration, a pipe capacity,
               or baseline message.";
            case telemetry-config {
              description
                "Used to set telemetry parameters such as setting
                 low, mid, and high percentile values.";
              container current-config {
                description
                  "Current telemetry configuration values.";
                uses telemetry-parameters;
                uses unit-config;
                leaf server-originated-telemetry {
                  type boolean;
                  description
                    "Used by a DOTS client to enable/disable whether it
                     accepts pre-or-ongoing-mitigation telemetry from
                     the DOTS server.";
                }
                leaf telemetry-notify-interval {
                  type uint32 {
                    range "1 .. 3600";
                  }
                  units "seconds";
                  description
                    "Minimum number of seconds between successive
                     telemetry notifications.";
                }
              }
            }
            case pipe {
              description
                "Total pipe capacity of a DOTS client domain.";
              list total-pipe-capacity {
                key "link-id unit";
                description
                  "Total pipe capacity of a DOTS client domain.";
                leaf link-id {
                  type nt:link-id;
                  description
                    "Identifier of an interconnection link of
                     the DOTS client domain.";
                }
                leaf capacity {
```

```
                    type uint64;
                    mandatory true;
                    description
                      "Pipe capacity. This attribute is mandatory when
                       total-pipe-capacity is included in a message.";
                  }
                  leaf unit {
                    type unit;
                    description
                      "The traffic can be measured using unit classes:
                       packets per second (pps), Bits per Second (bit/s),
                       and/or bytes per second (Byte/s).

                       For a given type, the DOTS agents auto-scale
                       to the appropriate units (e.g., megabit-ps,
                       kilobit-ps).";
                  }
                }
              }
              case baseline {
                description
                  "Traffic baseline information of a DOTS client domain.";
                list baseline {
                  key "id";
                  description
                    "Traffic baseline information of a DOTS client
                     domain.";
                  leaf id {
                    type uint32;
                    must '. >= 1';
                    description
                      "An identifier that uniquely identifies a baseline
                       entry communicated by a DOTS client.";
                  }
                  uses baseline;
                }
              }
            }
          }
        }
        case telemetry {
          description
            "Indicates the message is about telemetry.";
          list pre-or-ongoing-mitigation {
            description
              "Pre-or-ongoing-mitigation telemetry per DOTS client.
               The keys of the list are 'cuid' and 'tmid', but these
               keys are not represented here because these keys are
```

```
               conveyed as mandatory Uri-Paths in requests.
                Omitting keys is compliant with RFC8791.";
          choice direction {
            description
              "Indicates the communication direction in which the
               data nodes can be included.";
            case server-to-client-only {
              description
                "These data nodes appear only in a mitigation message
                 sent from the server to the client.";
              leaf tmid {
                type uint32;
                description
                  "An identifier to uniquely demux telemetry data sent
                   using the same message.";
              }
            }
          }
          container target {
            description
              "Indicates the target. At least one of the attributes
               'target-prefix', 'target-fqdn', 'target-uri',
               'alias-name', or 'mid-list' must be present in the
               target definition.";
            uses data-channel:target;
            leaf-list alias-name {
              type string;
              description
                "An alias name that points to a resource.";
            }
            leaf-list mid-list {
              type uint32;
              description
                "Reference a list of associated mitigation requests.";
            }
          }
          uses pre-or-ongoing-mitigation;
        }
      }
    }
  }
}
<CODE ENDS>
```

10.2. Vendor Attack Mapping Details YANG Module

```
<CODE BEGINS> file "ietf-dots-mapping@2020-06-26.yang"
module ietf-dots-mapping {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dots-mapping";
  prefix dots-mapping;

  import ietf-dots-data-channel {
    prefix data-channel;
    reference
      "RFC 8783: Distributed Denial-of-Service Open Threat
                 Signaling (DOTS) Data Channel Specification";
  }

  organization
    "IETF DDoS Open Threat Signaling (DOTS) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/dots/>
     WG List:  <mailto:dots@ietf.org>

     Author:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>

     Author:   Jon Shallow
               <mailto:supjps-ietf@jpshallow.com>";
  description
    "This module contains YANG definitions for the sharing
     DDoS attack mapping details between a DOTS client and
     a DOTS server, by means of the DOTS data channel.

     Copyright (c) 2020 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2020-06-26 {
    description
      "Initial revision.";
    reference
```

```
      "RFC XXXX: Distributed Denial-of-Service Open Threat
              Signaling (DOTS) Telemetry";
}

feature dots-telemetry {
  description
    "This feature indicates that DOTS telemetry data can be
     shared between DOTS clients and servers.";
}

grouping attack-mapping {
  description
    "A set of information used for sharing vendor attack mapping
     information with a peer.";
  list vendor {
    key "vendor-id";
    description
      "Vendor attack mapping information of the client/server";
    leaf vendor-id {
      type uint32;
      description
        "Vendor ID is a security vendor's Enterprise Number.";
    }
    leaf vendor-name {
      type string;
      description
        "The name of the vendor (e.g., company A).";
    }
    leaf last-updated {
      type uint64;
      mandatory true;
      description
        "The time the mapping table was updated. It is represented
         in seconds relative to 1970-01-01T00:00:00Z in UTC time.";
    }
    list attack-mapping {
      key "attack-id";
      description
        "Attack mapping details.";
      leaf attack-id {
        type uint32;
        description
          "Unique identifier assigned by the vendor for the attack.";
      }
      leaf attack-description {
        type string;
        mandatory true;
        description
```

```
              "Textual representation of attack description. Natural
               Language Processing techniques (e.g., word embedding)
               can possibly be used to map the attack description to
               an attack type.";
          }
        }
      }
    }

    augment "/data-channel:dots-data/data-channel:dots-client" {
      if-feature "dots-telemetry";
      description
        "Augments the data channel with a vendor attack
         mapping table of the DOTS client.";
      container vendor-mapping {
        description
          "Used by DOTS clients to share their vendor
           attack mapping information with DOTS servers.";
        uses attack-mapping;
      }
    }

    augment "/data-channel:dots-data/data-channel:capabilities" {
      if-feature "dots-telemetry";
      description
        "Augments the DOTS server capabilities with a
         parameter to indicate whether they can share
         attack mapping details.";
      leaf vendor-mapping-enabled {
        type boolean;
        config false;
        description
          "Indicates that the server supports sharing
           attack vendor mapping details with DOTS clients.";
      }
    }

    augment "/data-channel:dots-data" {
      if-feature "dots-telemetry";
      description
        "Augments the data channel with a vendor attack
         mapping table of the DOTS server.";
      container vendor-mapping {
        config false;
        description
          "Includes the list of vendor attack mapping details
           that will be shared upon request with DOTS clients.";
        uses attack-mapping;
```

```
      }
    }
  }
 <CODE ENDS>
```

11. YANG/JSON Mapping Parameters to CBOR

   All DOTS telemetry parameters in the payload of the DOTS signal
   channel MUST be mapped to CBOR types as shown in Table 2:

   o Note: Implementers must check that the mapping output provided by
      their YANG-to-CBOR encoding schemes is aligned with the content of
      Table 2.

Parameter Name	YANG Type	CBOR Key	CBOR Major Type & Information	JSON Type
tsid	uint32	TBA1	0 unsigned	Number
telemetry	container	TBA2	5 map	Object
low-percentile	decimal64	TBA3	6 tag 4 [-2, integer]	String
mid-percentile	decimal64	TBA4	6 tag 4 [-2, integer]	String
high-percentile	decimal64	TBA5	6 tag 4 [-2, integer]	String
unit-config	list	TBA6	4 array	Array
unit	enumeration	TBA7	0 unsigned	String
unit-status	boolean	TBA8	7 bits 20	False
			7 bits 21	True
total-pipe-capacity	list	TBA9	4 array	Array
link-id	string	TBA10	3 text string	String
pre-or-ongoing- mitigation	list	TBA11	4 array	Array
total-traffic-normal	list	TBA12	4 array	Array
low-percentile-g	yang:gauge64	TBA13	0 unsigned	String
mid-percentile-g	yang:gauge64	TBA14	0 unsigned	String
high-percentile-g	yang:gauge64	TBA15	0 unsigned	String
peak-g	yang:gauge64	TBA16	0 unsigned	String
total-attack-traffic	list	TBA17	4 array	Array
total-traffic	list	TBA18	4 array	Array
total-connection- capacity	list	TBA19	4 array	Array
connection	uint64	TBA20	0 unsigned	String
connection-client	uint64	TBA21	0 unsigned	String
embryonic	uint64	TBA22	0 unsigned	String
embryonic-client	uint64	TBA23	0 unsigned	String

connection-ps	uint64	TBA24	0 unsigned	String
connection-client-ps	uint64	TBA25	0 unsigned	String
request-ps	uint64	TBA26	0 unsigned	String
request-client-ps	uint64	TBA27	0 unsigned	String
partial-request-ps	uint64	TBA28	0 unsigned	String
partial-request- client-ps	uint64	TBA29	0 unsigned	String
total-attack- connection	container	TBA30	5 map	Object
low-percentile-l	list	TBA31	4 array	Array
mid-percentile-l	list	TBA32	4 array	Array
high-percentile-l	list	TBA33	4 array	Array
peak-l	list	TBA34	4 array	Array
attack-detail	list	TBA35	4 array	Array
id	uint32	TBA36	0 unsigned	Number
attack-id	uint32	TBA37	0 unsigned	Number
attack-description	string	TBA38	3 text string	String
attack-severity	enumeration	TBA39	0 unsigned	String
start-time	uint64	TBA40	0 unsigned	String
end-time	uint64	TBA41	0 unsigned	String
source-count	container	TBA42	5 map	Object
top-talker	container	TBA43	5 map	Object
spoofed-status	boolean	TBA44	7 bits 20	False
			7 bits 21	True
low-percentile-c	container	TBA45	5 map	Object
mid-percentile-c	container	TBA46	5 map	Object
high-percentile-c	container	TBA47	5 map	Object
peak-c	container	TBA48	5 map	Object
baseline	container	TBA49	5 map	Object
current-config	container	TBA50	5 map	Object
max-config-values	container	TBA51	5 map	Object
min-config-values	container	TBA52	5 map	Object
supported-unit-classes	container	TBA53	5 map	Object
server-originated- telemetry	boolean	TBA54	7 bits 20 7 bits 21	False True
telemetry-notify- interval	uint32	TBA55	0 unsigned	Number
tmid	uint32	TBA56	0 unsigned	Number
measurement-interval	enumeration	TBA57	0 unsigned	String
measurement-sample	enumeration	TBA58	0 unsigned	String
talker	list	TBA59	4 array	Array
source-prefix	inet: ip-prefix	TBA60	3 text string	String
mid-list	leaf-list	TBA61	4 array	Array
	uint32		0 unsigned	Number
source-port-range	list	TBA62	4 array	Array
source-icmp-type- range	list	TBA63	4 array	Array

target	container	TBA64	5 map	Object
capacity	uint64	TBA65	0 unsigned	String
protocol	uint8	TBA66	0 unsigned	Number
total-traffic- normal-per-protocol	list	TBA67	4 array	Array
total-traffic- normal-per-port	list	TBA68	4 array	Array
total-connection- capacity-per-port	list	TBA69	4 array	Array
total-traffic- protocol	list	TBA70	4 array	Array
total-traffic-port	list	TBA71	4 array	Array
total-attack- traffic-protocol	list	TBA72	4 array	Array
total-attack- traffic-port	list	TBA73	4 array	Array
total-attack- connection-port	list	TBA74	4 array	Array
port	inet: port-number	TBA75	0 unsigned	Number
query-type	leaf-list	TBA76	4 array	Array
			0 unsigned	String
vendor-id	uint32	TBA77	0 unsigned	Number
ietf-dots-telemetry: telemetry-setup	container	TBA78	5 map	Object
ietf-dots-telemetry: total-traffic	list	TBA79	4 array	Array
ietf-dots-telemetry: total-attack-traffic	list	TBA80	4 array	Array
ietf-dots-telemetry: total-attack- connection	container	TBA81	5 map	Object
ietf-dots-telemetry: attack-detail	list	TBA82	4 array	Array
ietf-dots-telemetry: telemetry	container	TBA83	5 map	Object
current-g	yang:gauge64	TBA84	0 unsigned	String
current-l	list	TBA85	4 array	Array
current-c	container	TBA86	5 map	Object
lower-type	uint8	TBA87	0 unsigned	Number
upper-type	uint8	TBA88	0 unsigned	Number

                 Table 2: YANG/JSON Mapping Parameters to CBOR

12. IANA Considerations

12.1. DOTS Signal Channel CBOR Key Values

This specification registers the DOTS telemetry attributes in the
IANA "DOTS Signal Channel CBOR Key Values" registry [Key-Map].

The DOTS telemetry attributes defined in this specification are
comprehension-optional parameters.

o Note to the RFC Editor: CBOR keys are assigned from the 128-255
   range.

Parameter Name	CBOR Key Value	CBOR Major Type	Change Controller	Specification Document(s)
tsid	TBA1	0	IESG	[RFCXXXX]
telemetry	TBA2	5	IESG	[RFCXXXX]
low-percentile	TBA3	6tag4	IESG	[RFCXXXX]
mid-percentile	TBA4	6tag4	IESG	[RFCXXXX]
high-percentile	TBA5	6tag4	IESG	[RFCXXXX]
unit-config	TBA6	4	IESG	[RFCXXXX]
unit	TBA7	0	IESG	[RFCXXXX]
unit-status	TBA8	7	IESG	[RFCXXXX]
total-pipe-capacity	TBA9	4	IESG	[RFCXXXX]
link-id	TBA10	3	IESG	[RFCXXXX]
pre-or-ongoing-mitigation	TBA11	4	IESG	[RFCXXXX]
total-traffic-normal	TBA12	4	IESG	[RFCXXXX]
low-percentile-g	TBA13	0	IESG	[RFCXXXX]
mid-percentile-g	TBA14	0	IESG	[RFCXXXX]
high-percentile-g	TBA15	0	IESG	[RFCXXXX]
peak-g	TBA16	0	IESG	[RFCXXXX]
total-attack-traffic	TBA17	4	IESG	[RFCXXXX]
total-traffic	TBA18	4	IESG	[RFCXXXX]
total-connection-capacity	TBA19	4	IESG	[RFCXXXX]
connection	TBA20	0	IESG	[RFCXXXX]
connection-client	TBA21	0	IESG	[RFCXXXX]
embryonic	TBA22	0	IESG	[RFCXXXX]
embryonic-client	TBA23	0	IESG	[RFCXXXX]
connection-ps	TBA24	0	IESG	[RFCXXXX]
connection-client-ps	TBA25	0	IESG	[RFCXXXX]
request-ps	TBA26	0	IESG	[RFCXXXX]
request-client-ps	TBA27	0	IESG	[RFCXXXX]
partial-request-ps	TBA28	0	IESG	[RFCXXXX]

partial-request- client-ps	TBA29	0	IESG	[RFCXXXX]	
total-attack- connection	TBA30	5	IESG	[RFCXXXX]	
low-percentile-l	TBA31	4	IESG	[RFCXXXX]	
mid-percentile-l	TBA32	4	IESG	[RFCXXXX]	
high-percentile-l	TBA33	4	IESG	[RFCXXXX]	
peak-l	TBA34	4	IESG	[RFCXXXX]	
attack-detail	TBA35	4	IESG	[RFCXXXX]	
id	TBA36	0	IESG	[RFCXXXX]	
attack-id	TBA37	0	IESG	[RFCXXXX]	
attack-description	TBA38	3	IESG	[RFCXXXX]	
attack-severity	TBA39	0	IESG	[RFCXXXX]	
start-time	TBA40	0	IESG	[RFCXXXX]	
end-time	TBA41	0	IESG	[RFCXXXX]	
source-count	TBA42	5	IESG	[RFCXXXX]	
top-talker	TBA43	5	IESG	[RFCXXXX]	
spoofed-status	TBA44	7	IESG	[RFCXXXX]	
low-percentile-c	TBA45	5	IESG	[RFCXXXX]	
mid-percentile-c	TBA46	5	IESG	[RFCXXXX]	
high-percentile-c	TBA47	5	IESG	[RFCXXXX]	
peak-c	TBA48	5	IESG	[RFCXXXX]	
ietf-dots-signal-cha	TBA49	5	IESG	[RFCXXXX]	
current-config	TBA50	5	IESG	[RFCXXXX]	
max-config-value	TBA51	5	IESG	[RFCXXXX]	
min-config-values	TBA52	5	IESG	[RFCXXXX]	
supported-unit-classes	TBA55	5	IESG	[RFCXXXX]	
server-originated- telemetry	TBA54	7	IESG	[RFCXXXX]	
telemetry-notify- interval	TBA55	0	IESG	[RFCXXXX]	
tmid	TBA56	0	IESG	[RFCXXXX]	
measurement-interval	TBA57	0	IESG	[RFCXXXX]	
measurement-sample	TBA58	0	IESG	[RFCXXXX]	
talker	TBA59	0	IESG	[RFCXXXX]	
source-prefix	TBA60	0	IESG	[RFCXXXX]	
mid-list	TBA61	4	IESG	[RFCXXXX]	
source-port-range	TBA62	4	IESG	[RFCXXXX]	
source-icmp-type- range	TBA63	4	IESG	[RFCXXXX]	
target	TBA64	5	IESG	[RFCXXXX]	
capacity	TBA65	0	IESG	[RFCXXXX]	
protocol	TBA66	0	IESG	[RFCXXXX]	
total-traffic- normal-per-protocol	TBA67	4	IESG	[RFCXXXX]	
total-traffic- normal-per-port	TBA68	4	IESG	[RFCXXXX]	
total-connection-	TBA69	4	IESG	[RFCXXXX]	

capacity-per-port				
total-traffic- protocol	TBA70	4	IESG	[RFCXXXX]
total-traffic-port	TBA71	4	IESG	[RFCXXXX]
total-attack- traffic-protocol	TBA72	4	IESG	[RFCXXXX]
total-attack- traffic-port	TBA73	4	IESG	[RFCXXXX]
total-attack- connection-port	TBA74	4	IESG	[RFCXXXX]
port	TBA75	0	IESG	[RFCXXXX]
query-type	TBA76	4	IESG	[RFCXXXX]
vendor-id	TBA77	0	IESG	[RFCXXXX]
ietf-dots-telemetry: telemetry-setup	TBA78	5	IESG	[RFCXXXX]
ietf-dots-telemetry: total-traffic	TBA79	0	IESG	[RFCXXXX]
ietf-dots-telemetry: total-attack-traffic	TBA80	0	IESG	[RFCXXXX]
ietf-dots-telemetry: total-attack- connection	TBA81	0	IESG	[RFCXXXX]
ietf-dots-telemetry: attack-detail	TBA82	4	IESG	[RFCXXXX]
ietf-dots-telemetry: telemetry	TBA83	5	IESG	[RFCXXXX]
current-g	TBA84	0	IESG	[RFCXXXX]
current-l	TBA85	4	IESG	[RFCXXXX]
current-c	TBA86	5	IESG	[RFCXXXX]
lower-type	TBA87	0	IESG	[RFCXXXX]
upper-type	TBA88	0	IESG	[RFCXXXX]

Table 3: Registered DOTS Signal Channel CBOR Key Values

Note that 'lower-type' and 'upper-type' are also requested for
assignment in the call-home I-D. Both I-Ds should be sync'ed as
depending the one that will make it first to the IANA.

12.2. DOTS Signal Channel Conflict Cause Codes

This specification requests IANA to assign a new code from the "DOTS
Signal Channel Conflict Cause Codes" registry [Cause].

```
+------+------------------+-----------------------+-------------+
| Code | Label            |     Description        |  Reference  |
+======+==================+=======================+=============+
| TBA  | overlapping-pipes | Overlapping pipe scope |  [RFCXXXX]  |
+------+------------------+-----------------------+-------------+
```

          Table 4: Registered DOTS Signal Channel Conflict Cause Code

12.3. DOTS Signal Telemetry YANG Module

   This document requests IANA to register the following URIs in the
   "ns" subregistry within the "IETF XML Registry" [RFC3688]:

          URI: urn:ietf:params:xml:ns:yang:ietf-dots-telemetry
          Registrant Contact: The IESG.
          XML: N/A; the requested URI is an XML namespace.

          URI: urn:ietf:params:xml:ns:yang:ietf-dots-mapping
          Registrant Contact: The IESG.
          XML: N/A; the requested URI is an XML namespace.

   This document requests IANA to register the following YANG modules in
   the "YANG Module Names" subregistry [RFC6020] within the "YANG
   Parameters" registry.

          name: ietf-dots-telemetry
          namespace: urn:ietf:params:xml:ns:yang:ietf-dots-telemetry
          maintained by IANA: N
          prefix: dots-telemetry
          reference: RFC XXXX

          name: ietf-dots-mapping
          namespace: urn:ietf:params:xml:ns:yang:ietf-dots-mapping
          maintained by IANA: N
          prefix: dots-mapping
          reference: RFC XXXX

13. Security Considerations

13.1. DOTS Signal Channel Telemetry

   The security considerations for the DOTS signal channel protocol are
   discussed in Section 11 of [I-D.ietf-dots-rfc8782-bis]. The
   following discusses the security considerations that are specific to
   the DOTS signal channel extension defined in this document.

   The DOTS telemetry information includes DOTS client network topology,
   DOTS client domain pipe capacity, normal traffic baseline and

connections capacity, and threat and mitigation information. Such
information is sensitive; it MUST be protected at rest by the DOTS
server domain to prevent data leakage.

DOTS clients are typically trusted devices by the DOTS client domain.
DOTS clients may be co-located on network security services (e.g.,
firewall) and a compromised security service potentially can do a lot
more damage to the network. This assumption differs from the often
held view that devices are untrusted, often referred to as the "zero-
trust model". A compromised DOTS client can send fake DOTS telemetry
data to a DOTS server to mislead the DOTS server. This attack can be
prevented by monitoring and auditing DOTS clients to detect
misbehavior and to deter misuse, and by only authorizing the DOTS
client to convey the DOTS telemetry for specific target resources
(e.g., an application server is authorized to exchange DOTS telemetry
for its IP addresses but a DDoS mitigator can exchange DOTS telemetry
for any target resource in the network). As a reminder, this is
variation of dealing with compromised DOTS clients as discussed in
Section 11 of [I-D.ietf-dots-rfc8782-bis].

DOTS servers must be capable of defending themselves against DoS
attacks from compromised DOTS clients. The following non-
comprehensive list of mitigation techniques can be used by a DOTS
server to handle misbehaving DOTS clients:

o The probing rate (defined in Section 4.5 of
   [I-D.ietf-dots-rfc8782-bis]) can be used to limit the average data
   rate to the DOTS server.

o Rate-limiting DOTS telemetry, including those with new 'tmid'
   values, from the same DOTS client defends against DoS attacks that
   would result in varying the 'tmid' to exhaust DOTS server
   resources. Likewise, the DOTS server can enforce a quota and
   time-limit on the number of active pre-or-ongoing-mitigation
   telemetry data (identified by 'tmid') from the DOTS client.

Note also that telemetry notification interval may be used to rate-
limit the pre-or-ongoing-mitigation telemetry notifications received
by a DOTS client domain.

13.2. Vendor Attack Mapping

The security considerations for the DOTS data channel protocol are
discussed in Section 10 of [RFC8783]. The following discusses the
security considerations that are specific to the DOTS data channel
extension defined in this document.

All data nodes defined in the YANG module specified in Section 10.2
which can be created, modified, and deleted (i.e., config true, which
is the default) are considered sensitive. Write operations to these
data nodes without proper protection can have a negative effect on
network operations. Appropriate security measures are recommended to
prevent illegitimate users from invoking DOTS data channel primitives
as discussed in [RFC8783]. Nevertheless, an attacker who can access
a DOTS client is technically capable of undertaking various attacks,
such as:

o Communicating invalid attack mapping details to the server
   ('/data-channel:dots-data/data-channel:dots-client/dots-
   telemetry:vendor-mapping'), which will mislead the server when
   correlating attack details.

Some of the readable data nodes in the YANG module specified in
Section 10.2 may be considered sensitive. It is thus important to
control read access to these data nodes. These are the data nodes
and their sensitivity:

o '/data-channel:dots-data/data-channel:dots-client/dots-
   telemetry:vendor-mapping' can be misused to infer the DDoS
   protection technology deployed in a DOTS client domain.

o '/data-channel:dots-data/dots-telemetry:vendor-mapping' can be
   used by a compromised DOTS client to leak the attack detection
   capabilities of the DOTS server. This is a variation of the
   compromised DOTS client attacks discussed in Section 13.1.

14. Contributors

The following individuals have contributed to this document:

o Li Su, CMCC, Email: suli@chinamobile.com

o Pan Wei, Huawei, Email: william.panwei@huawei.com

15. Acknowledgements

The authors would like to thank Flemming Andreasen, Liang Xia, and
Kaname Nishizuka co-authors of [I-D.doron-dots-telemetry] and
everyone who had contributed to that document.

The authors would like to thank Kaname Nishizuka, Wei Pan, and Yuuhei
Hayashi for comments and review.

Special thanks to Jon Shallow and Kaname Nishizuka for their
implementation and interoperability work.

Many thanks to Jan Lindblad for the yangdoctors review and Nagendra
Nainar for the opsdir review.

16. References

16.1. Normative References

[Enterprise-Numbers]
          "Private Enterprise Numbers", May 2020,
          <http://www.iana.org/assignments/enterprise-numbers.html>.

[I-D.ietf-dots-rfc8782-bis]
          Boucadair, M., Shallow, J., and T. Reddy.K, "Distributed
          Denial-of-Service Open Threat Signaling (DOTS) Signal
          Channel Specification", draft-ietf-dots-rfc8782-bis-06
          (work in progress), March 2021.

[I-D.ietf-dots-signal-filter-control]
          Nishizuka, K., Boucadair, M., Reddy, T., and T. Nagata,
          "Controlling Filtering Rules Using Distributed Denial-of-
          Service Open Threat Signaling (DOTS) Signal Channel",
          draft-ietf-dots-signal-filter-control-07 (work in
          progress), June 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
          DOI 10.17487/RFC3688, January 2004,
          <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
          the Network Configuration Protocol (NETCONF)", RFC 6020,
          DOI 10.17487/RFC6020, October 2010,
          <https://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
          RFC 6991, DOI 10.17487/RFC6991, July 2013,
          <https://www.rfc-editor.org/info/rfc6991>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
          Application Protocol (CoAP)", RFC 7252,
          DOI 10.17487/RFC7252, June 2014,
          <https://www.rfc-editor.org/info/rfc7252>.

   [RFC7641] Hartke, K., "Observing Resources in the Constrained
              Application Protocol (CoAP)", RFC 7641,
              DOI 10.17487/RFC7641, September 2015,
              <https://www.rfc-editor.org/info/rfc7641>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

   [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in
              the Constrained Application Protocol (CoAP)", RFC 7959,
              DOI 10.17487/RFC7959, August 2016,
              <https://www.rfc-editor.org/info/rfc7959>.

   [RFC7970] Danyliw, R., "The Incident Object Description Exchange
              Format Version 2", RFC 7970, DOI 10.17487/RFC7970,
              November 2016, <https://www.rfc-editor.org/info/rfc7970>.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
              <https://www.rfc-editor.org/info/rfc8040>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N.,
              Ananthakrishnan, H., and X. Liu, "A YANG Data Model for
              Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March
              2018, <https://www.rfc-editor.org/info/rfc8345>.

   [RFC8783] Boucadair, M., Ed. and T. Reddy.K, Ed., "Distributed
              Denial-of-Service Open Threat Signaling (DOTS) Data
              Channel Specification", RFC 8783, DOI 10.17487/RFC8783,
              May 2020, <https://www.rfc-editor.org/info/rfc8783>.

   [RFC8791] Bierman, A., Bjoerklund, M., and K. Watsen, "YANG Data
              Structure Extensions", RFC 8791, DOI 10.17487/RFC8791,
              June 2020, <https://www.rfc-editor.org/info/rfc8791>.

   [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object
              Representation (CBOR)", STD 94, RFC 8949,
              DOI 10.17487/RFC8949, December 2020,
              <https://www.rfc-editor.org/info/rfc8949>.

16.2. Informative References

   [Cause]   IANA, "DOTS Signal Channel Conflict Cause Codes",
              <https://www.iana.org/assignments/dots/dots.xhtml#dots-
              signal-channel-conflict-cause-codes>.

   [I-D.doron-dots-telemetry]
              Doron, E., Reddy, T., Andreasen, F., (Frank), L. X., and
              K. Nishizuka, "Distributed Denial-of-Service Open Threat
              Signaling (DOTS) Telemetry Specifications", draft-doron-
              dots-telemetry-00 (work in progress), October 2016.

   [I-D.ietf-core-new-block]
              Boucadair, M. and J. Shallow, "Constrained Application
              Protocol (CoAP) Block-Wise Transfer Options for Faster
              Transmission", draft-ietf-core-new-block-11 (work in
              progress), April 2021.

   [I-D.ietf-dots-multihoming]
              Boucadair, M., Reddy, T., and W. Pan, "Multi-homing
              Deployment Considerations for Distributed-Denial-of-
              Service Open Threat Signaling (DOTS)", draft-ietf-dots-
              multihoming-05 (work in progress), November 2020.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Moskowitz, R., Teague, N., Xia,
              L., and K. Nishizuka, "Use cases for DDoS Open Threat
              Signaling", draft-ietf-dots-use-cases-25 (work in
              progress), July 2020.

   [Key-Map] IANA, "DOTS Signal Channel CBOR Key Values",
              <https://www.iana.org/assignments/dots/dots.xhtml#dots-
              signal-channel-cbor-key-values>.

   [RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis,
              "Framework for IP Performance Metrics", RFC 2330,
              DOI 10.17487/RFC2330, May 1998,
              <https://www.rfc-editor.org/info/rfc2330>.

   [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
              BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
              <https://www.rfc-editor.org/info/rfc8340>.

   [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K.,
              and R. Wilton, "YANG Library", RFC 8525,
              DOI 10.17487/RFC8525, March 2019,
              <https://www.rfc-editor.org/info/rfc8525>.

   [RFC8612]   Mortensen, A., Reddy, T., and R. Moskowitz, "DDoS Open
               Threat Signaling (DOTS) Requirements", RFC 8612,
               DOI 10.17487/RFC8612, May 2019,
               <https://www.rfc-editor.org/info/rfc8612>.

Authors' Addresses

   Mohamed Boucadair (editor)
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

   Tirumaleswar Reddy (editor)
   McAfee, Inc.
   Embassy Golf Link Business Park
   Bangalore, Karnataka 560071
   India

   Email: kondtir@gmail.com

   Ehud Doron
   Radware Ltd.
   Raoul Wallenberg Street
   Tel-Aviv 69710
   Israel

   Email: ehudd@radware.com

   Meiling Chen
   CMCC
   32, Xuanwumen West
   BeiJing, BeiJing 100053
   China

   Email: chenmeiling@chinamobile.com

   Jon Shallow
   United Kingdom

   Email: supjps-ietf@jpshallow.com